ATTENDANCE MONITORING SYSTEM USING FACE RECOGNITION

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

KRITIKA PATHAK (191285)

Under the supervision of

Dr. MONIKA BHARTI

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, 173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Attendance Monitoring System using Face Recognition"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from February 2023 to May 2023 under the supervision of **Dr. Monika Bharti,** Assistant Professor (SG), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan.

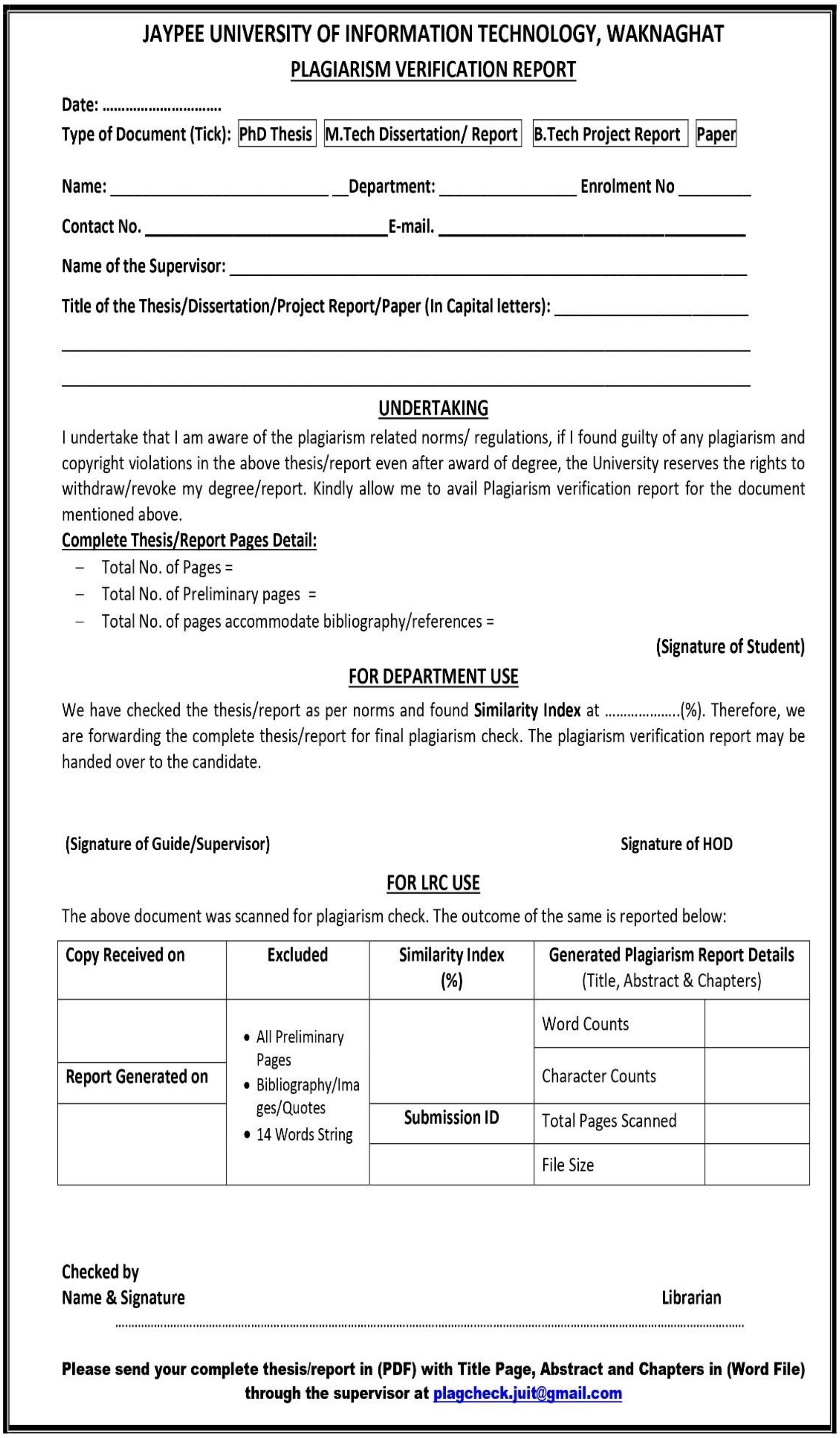
I also authenticate that I have carried out the above-mentioned project work under the proficiency stream of Cloud Computing.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Kritika Pathak [191285]

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Monika Bharti
Assistant Professor (SG),
Computer Science and Engineering
Dated



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………………

Type of Document (Tick): **PhD Thesis** **M.Tech Dissertation/ Report** **B.Tech Project Report** **Paper**

Name: _________________________ __**Department:** _______________ **Enrolment No** ________

Contact No. ___________________________**E-mail.** ___________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ______________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ………………..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

………

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, we would like to express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully
We are really grateful and wish our profound indebtedness to Dr. Vivek Kumar Sehgal,
HOD, Department of Computer Science and Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan.

Deep knowledge & keen interest of our supervisor, Dr. Monika Bharti in the field of "Machine Learning" helped us to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would also generously thank each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win.

We would also like to thank the various staff individuals, both teaching and non-teaching, for their convenient help and support throughout the working phase of the project.

Regards,

Kritika Pathak 191285
Computer Science and Engineering

TABLE OF CONTENT

LIST OF ABBREVIATIONS:

1. OpenCV : Open-Source Computer Vision
2. HOG: Histogram of Oriented Gradients
3. LBP: Local Binary Patterns
4. CNN: Convolutional neural networks
5. SVM: Support Vector Machine
6. KNN: k-Nearest Neighbors
7. ROC: Receiver Operating Characteristics
8. PIL: Python Imaging Library
9. SIFT: Scale Invariant Feature Transform

LIST OF FIGURES :

LIST OF TABLES :

CHAPTER-1
INTRODUCTION

1.1 Introduction

Attendance management is the process of efficient prevention of employee time theft by maintaining and tracking employee working hours, login and logout time. Organizations and Institutions use a number of techniques, from punching cards, maintaining spreadsheets to automated attendance systems by using Artificial Intelligence and Machine Learning Algorithms and biometric devices (Internet Of Things).

For effective functioning of an organization, maintaining a real time record is the key to improve the accuracy of the attendance system by Face recognition and real time attendance update. By combining the cloud and OpenCv technologies together. Using traditional roll-call methods was inefficient and time consuming. Use of updated and latest grown Artificial Intelligence and Machine Learning Algorithms make the attendance taking and recording system hassle free, convenient and less time consuming for the teachers and they could focus more on actually learning curve of students and delivering the best knowledge or getting the required outcome without worrying about proxies.

A staff attendance monitoring system is necessary for any organization, regardless of size, to effectively maintain tasks and projects. In order to manage differences and conflicts inside the organization, management must maintain records of each employee's time and attendance. Every technology has a significant improvement when mechanized in the modern, competitive environment. For a group of employees, manually tracking attendance is a time-consuming and ineffective operation. In this situation, a smart attendance management system with integrated facial recognition features will make a significant improvement.

This project comes as a solution to the challenges of time theft and error prone attendance data analysis faced by using the manual method. The instructor can feed the details of the student in the system by the code and then GUI of the application help the teacher to record the attendance of the student by face recognition mechanism and at last a csv file generated which contains the name time of each recorded student attention. This data is on a daily basis as spreadsheet data can be uploaded on the cloud database and can be accessed any time by the user and the admin. By registering the new user and entering their employee information into the source code, the administrator can add new users. Every employee's record may be viewed by the admin. A current record of the employee can be kept thanks to this method. Aim is to make the project accessible in web browsers as the project files and user database files will be stored and also hosted on the AWS cloud platform.

It was to record when workers entered and left the plant that the first employee time clocks were created in the late 19th century. The term "Time Card" originates from the fact that this mechanical time clock prints the employee working date and time into a paper card. With the use of the first-Time card, factory owners were able to keep track of the exact hours that each employee had put in. This safeguarded companies by ensuring that employees put in the required number of hours, and it insured the company employees by making it far more difficult for employers to cut down their working hours.

Due to the development of timekeeping technology, smaller electronic watches eventually supplanted huge mechanical ones. Now, workers had to precisely organize their time cards in the precise spot on the time cards that were produced, which had unique sections marked for clocking in and out. The 1990s saw the debut of industry-standard time clock software. As mechanical as well as electrical time clocks are likely to malfunction over their lifespan and are extremely expensive to replace or even repair, businesses started to move away from them around that time. The use of buddy punching is eliminated, productivity is increased, and labor expenses are decreased thanks to time clock software, which makes it simple for business owners to switch from time clock to payroll.

1.1.1 Old vs new ways of taking attendance

Some old ways to check Attendance:

A. **Traditional Time Clock:** This technique keeps track of attendance using a mechanical or electrical gadget. Data is recorded on paper punch cards, as the name implies, and is stamped and punched with the watch that is attached.

It allows for employee fraud. Without getting confirmation, workers can ask their coworkers to clock in and out.

B. **Paper timesheets:** This approach is incredibly ineffective in today's workplace. Employees still had to manually enter their working hours, and the reports they produced took a very long time to produce. Additionally, there is a chance for fraud because workers could make up activities only to complete their timesheets.

C. **Card Swipe:** Card Swipe is essentially a more contemporary method of timing. Employees can swipe into work with a magnetic card reader compatible with magnetic card readers. By enabling administrators to automatically generate timesheets and evaluate on-time performance, this technique saves time. However, this technique allows for employee fraud, just like conventional watches do. You can swipe without explicitly addressing your coworkers.

D. **Handwritten journals:** Handwritten journals are laborious and need to be completed by a responsible individual. Not very effective, hence most contemporary businesses no longer utilize it

E. **The Honor System:** The tenets of the honor system are not governed by any absolute laws. In essence, you have faith in your workers. Employees may, however, betray the confidence of the organization. Will these dated attendance techniques still function?
It may be argued that businesses all around the world continue to employ some of these techniques. But in today's fast-paced workplace, these outdated practices are both uncomfortable and ineffective.

A New Way to Record Attendance:

The way we track attendance has changed throughout the years as technology has grown. The most popular technique of verifying attendance in enterprises involves scanning an employee's fingerprint or palm print utilizing biometric identification systems, which use fingerprints as a biometric. The most recent technology is now more frequently used in the workplace. The workplace should make use of this technology for that reason. The most recent approach to keeping track of attendance, other than biometric devices like fingerprint sensors or face recognition techniques employing artificial intelligence and machine learning models.

Real time facial recognition:

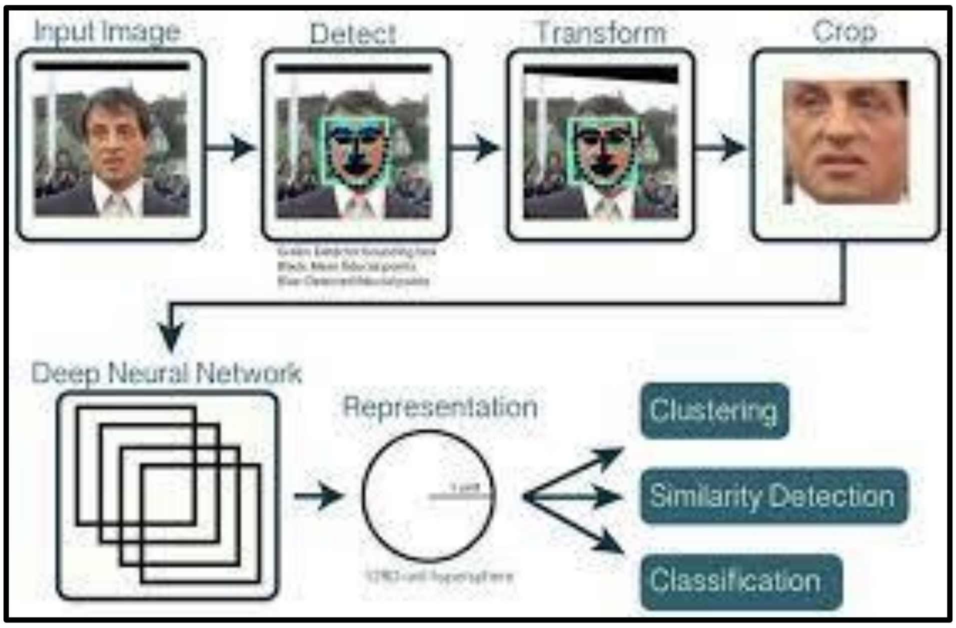


Figure 1.1.1.1: Flow of Facial Recognition working.

This project was completed with the aid of the OpenCV "Open-Source Computer Vision Library". Real-time applications and computing effectiveness were prioritized when developing OpenCV. For camera-based real-time facial identification, it is therefore perfect. Face Detection, Data Gathering, and Face Recognition are the common divisions of face recognition into these three stages.

The main goal is to recognize whether there are any human faces contained in an image, and if so, where exactly they are placed. Image windows are split into two categories for face detection: those with faces (which detract from the background) and those without (which are cluttered). It's challenging since, despite some visual resemblance, there may be considerable differences in terms of age, skin tone, and facial expression. The intricacy of the issue is increased by various lighting setups, image quality, and geometries, as well as the potential for partial occlusion and disguise.

You must first "capture" a face in order to recognize it. In the year 2001, Paul Viola along with Michael Jones published a study titled "Rapid Object Detection with a Boosted Cascade of Simple Features," which provided an effective approach for identification of different objects using “Haar feature-based cascade” classifiers. A cascade function is learned utilizing a significant number of both positive and negative photographs in this machine-learning approach. It is then used to locate things in other photos.

Human-face patches are extracted from photos after the detection of a face. The technique requires lots of pictures of faces (positive images) and pictures of objects without faces (negative images) to train this classifier. Both a detector and a trainer are included in OpenCV. OpenCV already comes with a large number of pre-trained classifiers for the face, eyes, grin, and other facial aspects.

The creation of a face database is necessary to achieve automatic recognition. Each person has multiple images taken of them, and the features are extracted and stored in the database. We'll just create a dataset where we'll keep a collection of grayscale images for each ID along with the area of each image that was used for the face.

1.2 Objectives

Utilizing the development of contemporary technology to create a clever system for quickly taking and managing attendance. Any type of organization needs a sophisticated attendance system, which is highly vital. Monitoring employee punctuality and managing absences are made easier with the use of an effective and reliable attendance system. A smart attendance system makes it possible to set up the attendance procedures and maintain accurate employee time-sheet validation.

The usage of a punch-in and punch-out system allows for the accurate tracking of the employee's working hours. Employee leaves might be watched over by managers in order to prevent absenteeism within the teams.

The payroll module interfaces with the smart attendance system to quickly and precisely calculate salaries as well as other benefits. This solution can be set up on smartphones and other devices, enabling administrators and supervisors to keep tabs on an employee's location as needed.

With its many functions, the smart attendance system streamlines and automates personnel scheduling. Within the organization, collaboration and communication between management and employees can be increased.

Employee attendance can be managed via facial recognition systems, which can quickly match faces and recognize human faces in milliseconds. The monitoring person is given access to a real-time control panel so they may check whether the faces match those in the saved image.

The smart attendance system is highly secure, and cost and time efficient as all the database and application is stored on high-speed, cost-efficient cloud servers.

1.3 Problem Statement

The traditional method of taking attendance has a major drawback in that it necessitates a lot of time-consuming, ineffective human interaction. The solution to these problems lies in the creation of intelligent and autonomous systems that can efficiently record, manage, and report attendance without the need for human intervention.

Recent technological advancements in areas like cloud computing, the internet of things, biometric sensors, and extremely effective near field communication devices have paved the way for the creation of systems that can operate with little human error and interference seven days a week.
For ensuring that both students and staff in various organizations maintain discipline, an attendance system is both a valuable and crucial tool.

The amount of work that employees perform is directly correlated with the length of time they spend at the office. in order for you to be aware of how they perform within the company

Monitor employee attendance at work to determine billing hours.
You need to be aware of employee work hours. It aids in the computation of your employees' workdays. This will help you know what salary you should give him/her.
Direct Costs – These are costs incurred directly as a result of employee absences. These costs include paid vacation, overtime, and replacement-worker costs.
Overhead Costs – These are costs incurred indirectly as a result of employee absences. Overhead costs include lost productivity and work delays due to employee absenteeism.

Businesses that use shift work typically need to know their absence rates. You can determine your employees' absenteeism rate by monitoring employee attendance. This enables management to plan when to hire a replacement for the position and determine any additional expenditures that might be required later.

Time and Attendance keeps track of absences and time spent in the office.
Both managers and employees have access to this information, which is openly disclosed. Employees are able to see how many vacation days they have left thanks to this. Additionally, it assists management in determining how much paid and unpaid time off to provide workers depending on their present attendance.

In an educational institution, these systems can be used as a tool:

- Against student proxies in classrooms.
- For maintaining attendance records for students and staff.
- For taking remote attendance effectively.

Because of masks and the contagious nature of the virus during the Covid-19 epidemic, implementing biometric verification methods like Face Recognition or Fingerprint sensing was not a practical option.
It became necessary to create an attendance system based on contactless authentication as a result.

1.4 Methodology

The manual and time-consuming process of recording attendance is the key issue that has to be addressed in this project. Traditional methods, such as paper-based or card-based systems, are prone to errors and require significant effort to maintain. The objective is to develop an automated attendance management system using face detection and face recognition functionalities to streamline the attendance process and improve accuracy.

The system needs to capture video input from a camera, detect and recognize faces in real-time, store attendance records securely, and provide a user-friendly interface for interaction. It should accurately identify individuals and record their attendance with timestamps. The system should also be scalable to handle many users and maintain high accuracy even in varying lighting conditions.

The system architecture involves integrating different components. It includes a video capture module to obtain live video input, a face detection module to identify faces in the video stream, a face recognition module to match faces with known identities, and a data storage module to store attendance records. The user interface should provide options to start the attendance recording process and view attendance records.

The implementation phase involves developing and integrating the different modules. It includes writing code to capture video using OpenCV, implementing face detection and recognition algorithms using libraries like dlib or OpenCV, and designing a database or file-based storage system to store attendance records. The implementation should ensure high accuracy, real-time performance, and robustness against various environmental factors.

Attendance system needs to undergo rigorous testing to ensure its functionality, accuracy, and reliability. Various test scenarios should be executed to evaluate its performance under different conditions, such as different lighting, angles, and facial variations. The system should be evaluated against a ground truth to measure its accuracy in identifying individuals and recording attendance correctly.

Once the system is thoroughly tested and evaluated, it can be deployed in the intended environment. Adequate documentation and user manuals should be provided to facilitate easy usage. Regular maintenance and updates should be carried out to address any bugs, security vulnerabilities, or performance improvements.

Training sessions should be conducted to familiarize users with the attendance system and its features. User support channels, such as documentation, FAQs, and a helpdesk, should be established to assist users in case of any issues or queries.

1.4.1 Facial Recognition using machine learning

The topic of machine learning-based face recognition is one that has made great strides in recent years. It involves applying computer techniques and algorithms to automatically recognize and validate people based on their face traits. Face recognition has attracted a lot of attention and found applications in a variety of fields, including security, surveillance, biometrics, and personalization. This is due to the expanding availability of high-resolution cameras and the growing demand for reliable and effective identification systems.

The process of face recognition using machine learning can be divided into several major steps. Let's know each of these steps in detail:

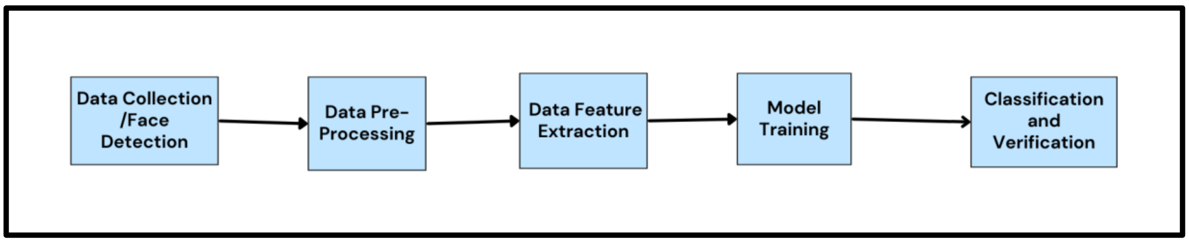

Data Collection /Face Detection
Data Pre-Processing
Data Feature Extraction
Model Training
Classification and Verification

Figure 1.4.1.1: Flowchart for steps involved in Facial Recognition through Machine Learning.

1. Data Collection: A dataset of face images must be compiled before developing a face recognition system. This dataset is anticipated to contain a large variety of people in varied poses, backdrops, lighting, and lighting settings, as well as different facial expressions. For data gathering, it is possible to use already-existing facial image databases or take photographs using cameras.

2. Preprocessing: To enhance the quality of the photos and standardize them for later analysis, the dataset undergoes preprocessing after it is obtained. One example of a preprocessing approach is face detection, which locates and extracts faces from the photos; normalization, which aligns and resizes the faces; and noise reduction, which gets rid of any artifacts or faults.

3. Feature Extraction: The extraction of useful and discriminative elements from the facial photographs is the most crucial step in face recognition. The distinctive traits of each person's face should be correctly reflected by these features. Traditional methods included manually made features like Local Binary Patterns (LBP) as well as Histogram of Oriented Gradients (HOG). Also, some deep learning techniques recently showed astonishing success in automatically learning discriminative features from raw face pictures. Convolutional neural networks (CNNs) are commonly used to extract deep features from photographs of faces.

4. Model Training: A machine learning model is taught to recognize and distinguish between different people after the features have been retrieved. a number of machine learning techniques, including neural networks, support vector machines (SVM), and k-nearest neighbors algorithm (k-NN), can be utilized for face identification. The train process entails feeding or delivering the extracted features into the model that is being utilized, together with their respective matching labels (identities), in order to determine the patterns and correlations between various faces.

5. Classification and Verification: Face recognition tasks can be carried out with the model once it has been trained. By assigning a face to a specific class or label, the algorithm may be able to predict how a face will be recognized during the classification process. In circumstances where the goal is to pick out specific individuals from a known group of people, this is useful. In the case of verification, the computer determines whether or not a specific face is the face of a specific person. This is useful when the system has to authenticate people based on the features of their faces.

6. Performance Evaluation: Accuracy, precision, recall, and receiver operating characteristic (ROC) curves are only a few of the metrics frequently used to judge the effectiveness of face recognition systems. A new dataset is used to test the system's performance, and its accuracy in recognizing or validating faces is measured. Additionally, the system's resistance to

changes in lighting, position, and expression is assessed. Performance evaluation is useful for system tuning and determining whether a system is suitable for usage in real-world scenarios.

7. Future Challenges: Face recognition using machine learning has advanced significantly in recent years, largely due to the availability of large datasets and powerful processing power. One type of deep learning model, convolutional neural networks (CNNs), has shown improved performance in learning accurate and discriminative face representations. Other techniques that have aided in the improvement of face recognition systems' reliability and accuracy include face alignment, 3D face modeling, and ensemble learning. The handling of occlusions and disguises, addressing changes in lighting, position, and expression, as well as protecting privacy and ethical considerations, continue to be problems.

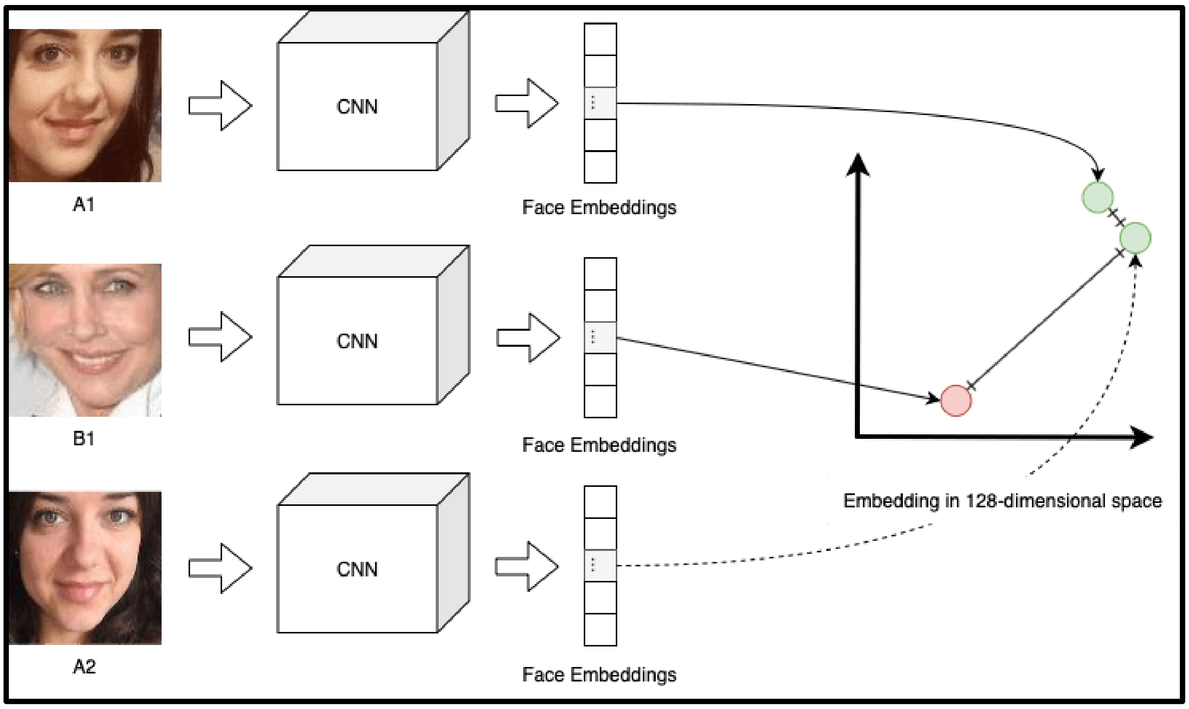


Figure 1.4.1.2: Represent how CNN layer and Face Embeddings work.

In conclusion, face recognition using machine learning has revolutionized the way we identify and verify individuals based on their facial features or face. Now the Facial features are the parameters to identify or recognize anyone when loaded in the system. We can use the growing Artificial Intelligence and Machine Learning field in various use cases, such as finding a person who got lost by using face recognition and making it a lot easier for cops to work, similarly Identifying the hackers or crime accounts on social media platforms, detecting an accident driver who is on the run. Nowadays AI power and human power run hand in hand and help to maintain the safety and ease to do things in a much better way even for a naive person.

1.4.2 Python module: Face_recognition

Face detection and face_recognition is not the same. In the face identification challenge, we identified the identities of each face detected, whereas in face detection, we had just identified the locations of human faces.

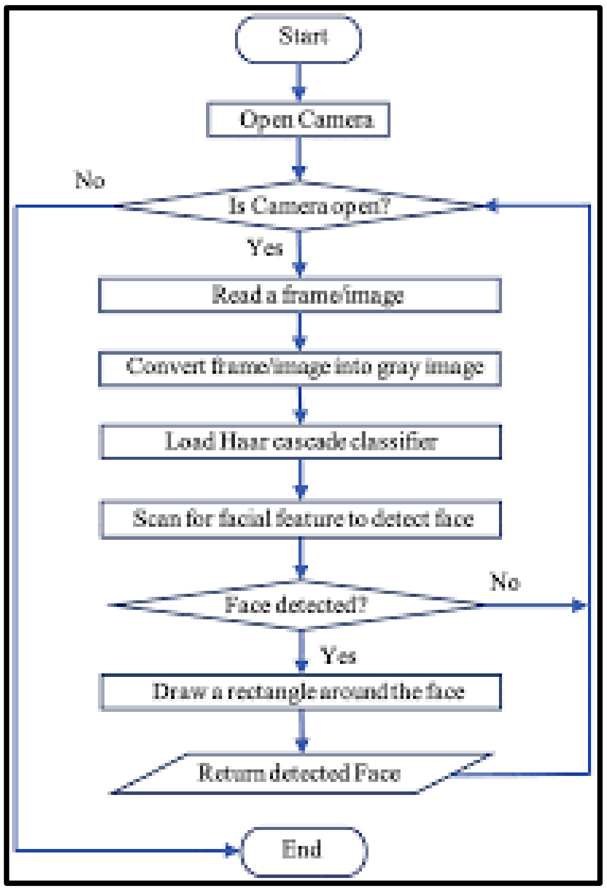

Start
Open Camera
No
Is Camera open?
Yes
Read a frame/image
Convert frame/image into gray image
Load Haar cascade classifier
Scan for facial feature to detect face
No
Face detected?
Yes
Draw a rectangle around the face
Return detected Face
End

Figure 1.4.2.1: Flowchart of Face recognition module.

Using face recognition Python module is a strong library that offers straightforward but efficient facial recognition capabilities. It is constructed on top of the highly optimized C++ library dlib, which is well-known for its effectiveness in computer vision tasks. The goal of the face_recognition module is to make the process of face detection, facial feature extraction, and face comparison simple so that both novice and seasoned developers may use it.

One of the main characteristics of the face_recognition module is the ability to recognize faces in still or moving photos. It uses a pre-trained deep learning algorithm to accurately recognize faces even in challenging circumstances including varying illumination, different poses, and partial occlusions. As a result, it is incredibly robust and reliable in a range of real-world situations.

The module enables facial landmark detection in addition to face detection, which distinguishes crucial facial features like the mouth, nose, and eyebrows. These landmarks are necessary for tasks like analyzing facial expressions, estimating poses, and face alignment. By accurately identifying facial landmarks, the module offers a more detailed study of facial traits and a better understanding of facial dynamics.

Another interesting feature is the face encoding capability of the face_recognition module. It makes use of face encodings, which are numerical representations of faces that are retrieved from seen faces using deep learning models. Since they capture the unique features of each face, these face encodings can be used for face comparison and recognition.

By calculating the similarity of face encodings, the module can determine whether two faces belong to the same person, enabling precise face recognition functionality.

Due to its compatibility with other well-known Python libraries like OpenCV and PIL (Python Imaging Library), the face_recognition module is simple to incorporate into workflows for the processing of images and videos. Programmers may manage different media formats, develop face recognition apps in real time, and incorporate the module's functionality into already started projects thanks to its integration.

All things considered, the face recognition module provides a simple and efficient way to carry out facial recognition operations, allowing programmers to easily add potent face detection, facial feature extraction, and face comparison capabilities to their applications.

1.4.3 Old Used Face Recognition Algorithm

The standards for facial recognition in today's world are not met by traditional face recognition algorithms. They were created with outdated, traditional face recognition algorithms.

Traditional facial recognition algorithms are available in OpenCV.

- **Eigen Faces:** As contrast to employing parts-based or feature-based methods, the face recognition technique known as Eigenfaces uses an appearance-based strategy to encode and compare images of individual faces holistically. Eigenfaces compares and encodes the variances between the photos by looking for differences among a set of portraits of human faces. In greater detail, an image with N pixels is viewed as a point (or vector) in N-dimensional space, and the eigenfaces are the primary factors of a distribution of faces, or alternatively, the eigenvectors of the covariance matrix of the collection of images of faces. Turk and Pentland (Turk and Pentland, 1991) used the Sirovich and Kirby (Sirovich and Kirby, 1987) principal component model to detect and identify human faces.

  The Eigenface approach, which served as the model for one of the best-selling products in the market, is widely regarded as the first practical facial recognition system. The original strategy has undergone multiple changes since it was created and published, and automatic facial recognition systems have seen many new advancements. Eigenfaces are still often employed as a baseline comparison approach to demonstrate the least projected performance of such a system.

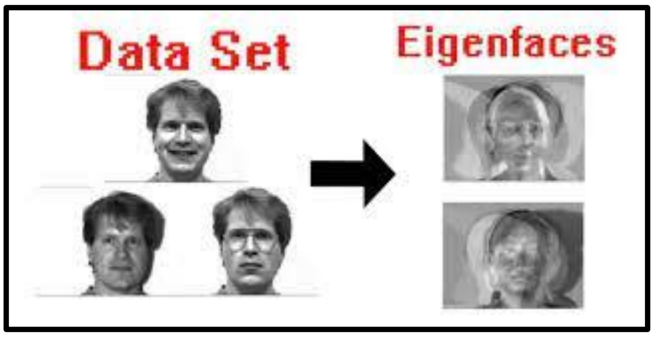


Figure 1.4.3.1: Depict how Eigenfaces are generated through a dataset.

There are two main points to consider:

1. Extracting the pertinent facial data, which might or might not be related to human perception of characteristics of the face such as the eyes, lips, nose, ears, and forehead. Capturing the statistical diversity between facial photos is one technique to do this.
2. effectively depict facial images. Each facial image can be represented by a limited number of characteristics, which lowers the processing and space complexity.

The eigenfaces can be explained as a group of characteristics that describe the overall pattern of variations among facial images. Then, a subset of the eigenfaces, that are linked to the biggest eigenvalues, are used to represent each face picture approximately. The majority of the variance in the training set is explained by these features.

Using Eigenfaces in Face Processing:

By selecting the subset of eigenvectors **U = u1, u2, u3, ,…., um** linked to the **m** largest eigen-values, the eigenfaces would span across an m-dimensional subspace of the initial picture space. As illustrated in figure below, this produces the so-called face space, whose axes are the eigenfaces and whose origin is the average face. One can calculate the distance in between or from the facial space in order to detect and recognize faces.

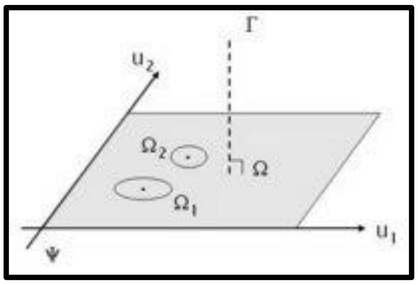


Figure 1.4.3.2: 2D face visualization of a space, with each axis representing two Eigenfaces.

Face detection

Face detection can be defined as identifying picture patches near to the face space since the facial space specifies the space of face images. To put it another way, the projection distance must fall inside a certain range. Of threshold value θ δ

The point-to-space distance, which is calculated as, is the distance between the face picture and its facial projection onto the face space.

$\delta = \| (\mathbf{I} - \mathbf{U}\hat{}\mathbf{U}\hat{}\top)(\Gamma - \Psi) \|$

To define the above, **I** stand for the identity matrix.
In general, using the above principle, the distance between an image and its face space projection is considerably smaller for a face than for a nonface object like a cup, bed, table, bottle, etc.

Face recognition

The expression =U() projects a new face into the face space, where U is the collection of important eigenvectors. Keep in mind that the new face is represented in face space by the weight vector Ω. The Euclidean distance can be minimized as a quick and easy method of identifying which face class Γ belongs to.

$\epsilon k = \| \Omega - \Omega k \|$

where k represents the kth face class in terms of weight vector Ωk, and k. The face is categorized as unknown if the minimum k is greater than a certain threshold θϵ and is believed to belong to class k otherwise. Above given figure helps us illustrate the projection and recognition by visualizing face space as a plane.

- S**cale-invariant feature transform** (**SIFT**) : It is a computer vision algorithm to detect, process, describe and map local features in images, invented by David Lowe in 1999. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally, the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

Stages involved in Scale Invariant Feature transform algorithm:

1. Scale-invariant feature detection: A large number of feature vectors that are resilient to local geometric distortion and invariant to changes in lighting as well as translation, scaling, and rotation of the source image are created from an image using the Lowe method for generating image features. These properties are shared by neurons in the monkey primary visual cortex that encode basic forms, colors, and movements for object recognition. The maxima and minima of a Gaussian function difference computed in scale space on a set of resampled and smoothed images represent the key points. Candidate sites with low contrast and edge response points

along an edge are ignored. Dominant orientations have been assigned to localized critical locations. As a result of these actions, the crucial locations will be more stable for matching and identification. Then, SIFT descriptors resistant to local affine distortion are produced by taking into consideration pixels within a radius of the significant location, blurring, and resampling local picture orientation planes.

2. Matching features and indexing: Indexing involves saving SIFT keys and finding corresponding keys in the new picture. The closest neighbors can be identified with high probability using a small amount of work thanks to a variation of the k-d tree technique employed by Lowe termed the best-bin-first search approach. Bins in feature space are searched in the order of their proximity to the query location using a modified search ordering for the k-d tree method that is used by the BBF algorithm. For effective search order determination, this search order requires the use of a heap-based priority queue. The closest neighbor of each key point in the database of keypoints from training photos is deemed to be the best potential match for that key point. The keypoints with the shortest Euclidean distance from the specified descriptor vector are referred to as the closest neighbors. By dividing the distance from the closest neighbor by the distance from the next-closest neighbor, one can calculate the likelihood that a match is accurate.

   Lowes discarded all matches with a distance ratio greater than 0.8, eliminating 90% of erroneous matches while tossing out less than 5% of true matches. The search was stopped after examining the first 200 nearest neighbor candidates in order to increase the effectiveness of the best-bin-first technique. For a database of 100,000 keypoints, this offers a speedup over exact closest neighbor search of around 2 orders of magnitude with less than a 5% reduction in the number of accurate matches.

3. Identification of clusters by Hough transform voting: To find keys that concur on a specific model pose, the Hough transformation is utilized to cluster trustworthy hypotheses models. By employing each feature to cast a vote for all object postures that are consistent with the feature, the Hough transform finds clusters of features by consistently interpreting them. The likelihood that the interpretation is accurate is significantly higher when groups of features are found to support a particular pose of

an item than when just one characteristic does. The location, orientation, and scale of the model are predicted from the match hypothesis by implementing a hash table for the entries. All clusters of at least three items in a bin are found using a search of the hash table, and the bins are then sorted by decreasing size.

Each SIFT key point is characterized in terms of its 2D location, scale, and orientation, and each matched key point in the database includes a record of its parameters in respect to the training image it was found in. The similarity transform implied by these four parameters only approximates the entire 6 degree-of-freedom pose space for a 3D object and does not account for any non-rigid deformations. With a scale factor of 2, 0.25 times the highest planned training picture dimension, and broad bin widths of 30 degrees for orientation, Lowe used these techniques. The SIFT key samples made at the larger scale weigh twice as much as the samples made at the smaller scale. Additionally, by giving the least-noisy scale more weight, this enhances recognition performance. To get around the problem of boundary effects in bin assignment, each key point match votes for the two closest bins in each dimension, giving each hypothesis a total of 16 entries and extending the range of viable poses.

4. Verification of the model by linear least squares: The parameters of the affine transformation linking the model to the image are solved using a linear least squares method, and each found cluster is then put through a verification process. The following is a possible way to express the transformation of a point in the model [x y]T to an image point [u v]T.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

where the model translation is [tx ty]T, the affine rotation, scale, and stretch are given by the parameters m1, m2, m3, and m4, respectively. It is possible to rewrite the previous equation to group the unknowns into a column vector and utilize that vector to solve for the transformation parameters.

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & & & & & \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ tx \\ ty \end{bmatrix} = \begin{bmatrix} u \\ v \\ . \\ . \end{bmatrix}$$

The above equation only shows one match, but any further matches in the future can be added, each of which adds two rows additionally to the first and the last matrices. A solution must be offered with a minimum of 3 matches. This linear system can be expressed as:

$$A\hat{\mathbf{x}} \approx \mathbf{b},$$

When x is an n-dimensional parameter vector that is unknown, b is a m-dimensional measurement vector that is predefined and known, and A is a known mXn matrix (with m > n).

The minimizing vector is as a result a solution of the normal equation.

$$A^T A\hat{\mathbf{x}} = A^T \mathbf{b}.$$

In terms of the matrix, the system of linear equations is solved. Known as the pseudoinverse of A, (AT)-1AT

$$\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}.$$

This reduces the total squared distance between the final model places and the corresponding associated image locations.

5. Detection of Outliers: Outliers can be removed after the parameter solution has been discovered by assessing whether each picture attribute and the model agree. Each match must be within half the error range that was assigned to the parameter in the Hough transform bins and agree with the linear least squares solution. The remaining points are used to find the minimal squares linear solution once the outliers have been removed, and the process is repeated. If fewer than three points remain after eliminating outliers, the match is ruled illegitimate. Additionally, a top-down comparing phase is used to incorporate any additional matches that agree with the expected model's position that the Hough transform bin may have missed due to inaccurate similar transform estimation or other issues.

Whether to accept or reject a model hypothesis is decided using a sophisticated probabilistic model. First, the projected model size, the number of features in the region, and the fit accuracy are used to calculate the predicted number of wrong matches to the model pose. A Bayesian probability estimation then provides the likelihood that the topic is present based on the true number of matching features found. A model is said to be legitimate if its ultimate probability of being correctly understood is greater than 0.98. Lowe's SIFT-based object recognition performs admirably, with the exception of conditions involving high illumination fluctuations and non-rigid transformations.

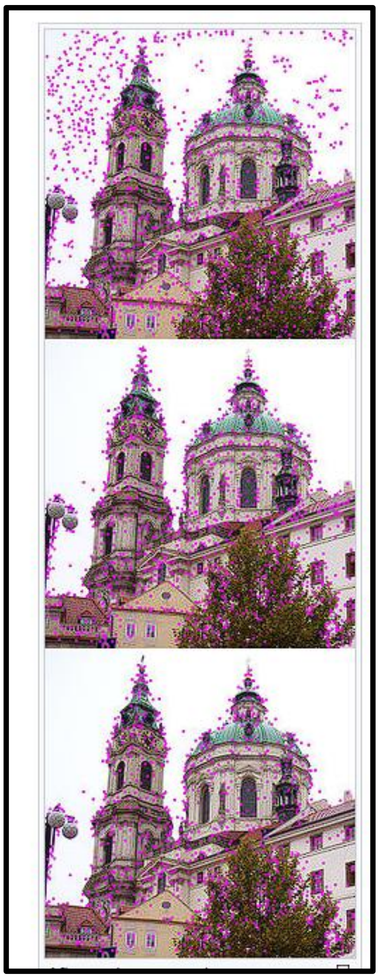


Figure 1.4.3.3: SIFT Algorithm application on image step by step.

The SIFT method first identifies scale space extrema (the locations of which are indicated in the top image), discards lower-contrast keypoints (the points that remain are displayed in the middle image), and then filters out the outer edges. The final graphic displays the key points that were generated. The upper image is an impactful way to represent how the SIFT Algorithm works.

1.4.4 The Algorithm used in current days:

- Local Binary Pattern histogram : These approaches differ in how they match input and output images and extract image data. The LBPH algorithm is a straightforward but extremely effective technique that is still used; however it is slow compared to newer algorithms.

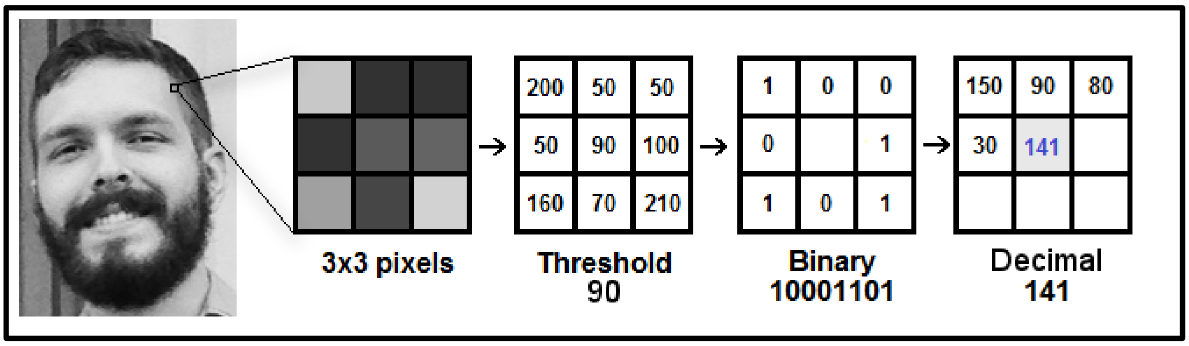


Figure 1.4.4.1: Represents face into pixels, threshold, binary and decimal.

- Face Net: FaceNet is a technique that can be used for face identification, verification, and grouping. It depends on a convolutional neural network (CNN). FaceNet operates by projecting facial images onto an area of Euclidean space, where the closer two images are to one another, the more comparable they are perceived to be. Images that have been cropped, resized, and altered around the facial area are used to train FaceNet.

  FaceNet, in contrast to earlier methods, derives mappings from the photos and produces embeddings directly without the need for a separate layer for identification or verification. The model's incredibly small size—only 128 bytes of data are used to describe each face—is a significant benefit. Researchers evaluated FaceNet on the

LFW and YouTube Faces DB in the study, attaining accuracy of over 95% and a 30% reduction in error rate when compared to the best prior result.

- ArcFace: An ML model called ArcFace aims to distinguish among a number of previously established classes. Then, a feature space is extracted using a backbone trained with ArcFace, allowing for downstream tasks like face verification and identification. Applications for face discovery and recognition can benefit from it. By learning distance measurements, ArcFace uses Similarity Learning (SL) to back the solution of classification tasks. To determine the separation between face images, angular margin loss is used in place of Softmax loss. The numerator and denominator of the loss function can be divided into two distinct components. Since we are minimizing the loss and the function of loss is negative, we would like to increase the nominator absolute values and decrease the denominator pure values :

1. The similarity of cosines between the normalized class embedding data and the class weight in the numerator is calculated using an inner product between the two vectors. The cosine similarity will be closer to 1 and closer to 0 depending on how close the two vectors are to collinearity. As a result, the larger our nominator, smaller the angles between the two vectors, and less our loss.
2. We want to reduce the relation between the cosines among our class instance and the weights of all the other classes in the denominator.

As a result, we obtain a loss term that requires separation from all other classes and proximity to the class mean.

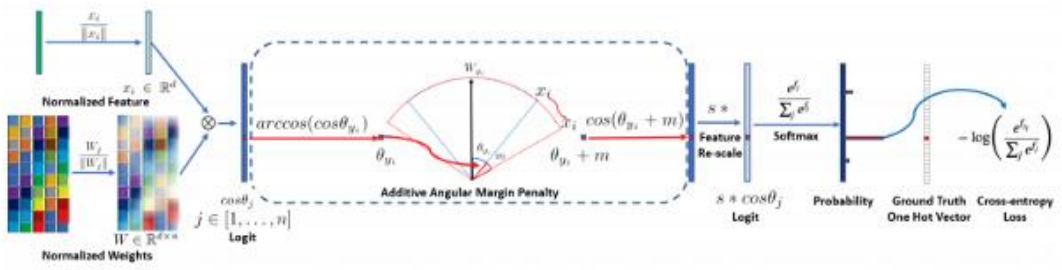


Figure 2. Training a DCNN for face recognition supervised by the ArcFace loss. Based on the feature x_i and weight W normalisation, we get the $\cos\theta_j$ (logit) for each class as $W_j^T x_i$. We calculate the $arccos\theta_{y_i}$ and get the angle between the feature x_i and the ground truth weight W_{y_i}. In fact, W_j provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θ_{y_i}. After that, we calculate $\cos(\theta_{y_i} + m)$ and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross entropy loss.

Algorithm 1 The Pseudo-code of ArcFace on MxNet

```
Input: Feature Scale $s$, Margin Parameter $m$ in Eq. 3, Class Number $n$, Ground-Truth ID $gt$.
  1. x = mx.symbol.L2Normalization (x, mode = 'instance')
  2. W = mx.symbol.L2Normalization (W, mode = 'instance')
  3. fc7 = mx.sym.FullyConnected (data = x, weight = W, no_bias = True, num_hidden = n)
  4. original_target_logit = mx.sym.pick (fc7, gt, axis = 1)
  5. theta = mx.sym.arccos (original_target_logit)
  6. marginal_target_logit = mx.sym.cos (theta + m)
  7. one_hot = mx.sym.one_hot (gt, depth = n, on_value = 1.0, off_value = 0.0)
  8. fc7 = fc7 + mx.sym.broadcast_mul (one_hot, mx.sym.expand_dims (marginal_target_logit - original_target_logit, 1))
  9. fc7 = fc7 * s
Output: Class-wise affinity score $fc7$.
```

Figure 1.4.4.2: Algorithm steps for Arcface.

The FC layer (completely connected) uses the inner products of weight and features in classification tasks, applying Softmax to the output after computing the features.

ArcFace produces an embedding space with enough space between several classes. Classes are better segregated as the embedded space becomes sparser.

- face.evoLVE : face.evoLVE The main application of the widely used and actively developing open-source library evoLVE is frontal face recognition. It offers all essential elements of face analytics, such as:
    1. Facial Position
    2. Data processing Foundations
    3. Loss mechanisms
    4. Adjustments to raise performance

It supports parallel-GPU training using PyTorch and PaddlePaddle, offers a variety of deep learning approaches for face recognition, and makes it simple to work with both massive datasets and low-shot databases with sparse data.

The images of common face benchmark datasets, both before and after alignment, are another crucial component of evoLVE. This makes it much simpler to assess models created by library users.

- OpenFace: For computer vision researchers creating apps based on facial analysis and recognition, OpenFace is a useful tool. The following kinds of face analysis skills are offered by it:
    1. Based on the article "Convolutional Experts Constrained Local Model" (Zadeh et al., 2017), facial landmark detection.
    2. Based on the publication "Cross-Dataset Learning and Person-Specific Normalization for Automatic Action Unit " (Baltrusaitis, et al., 2015), facial action unit recognition.
    3. Based on the study "Rendering of Eyes for Eye-Shape Registration and Gaze Estimation" (Wood et al., 2015), eye gaze estimation.
    4. estimation of a head stance.

Importantly, the toolset supports input from a regular webcam and is designed for real-time performance.

1.4.5 Deep Learning For Face Recognition

Deep convolutional neural networks (CNNs), which have received the greatest attention in the field of face recognition, are among the many deep learning techniques that have been developed. Deep learning has transformed facial recognition research since the introduction of the DeepFace and DeepID techniques in 2014. Since then, deep face recognition algorithms have significantly improved state-of-the-art performance and given rise to a huge variety of useful real-world applications. These algorithms learn selective face representation using a hierarchical design. Deep learning uses many processing layers to learn representations of data with different levels of feature extraction.

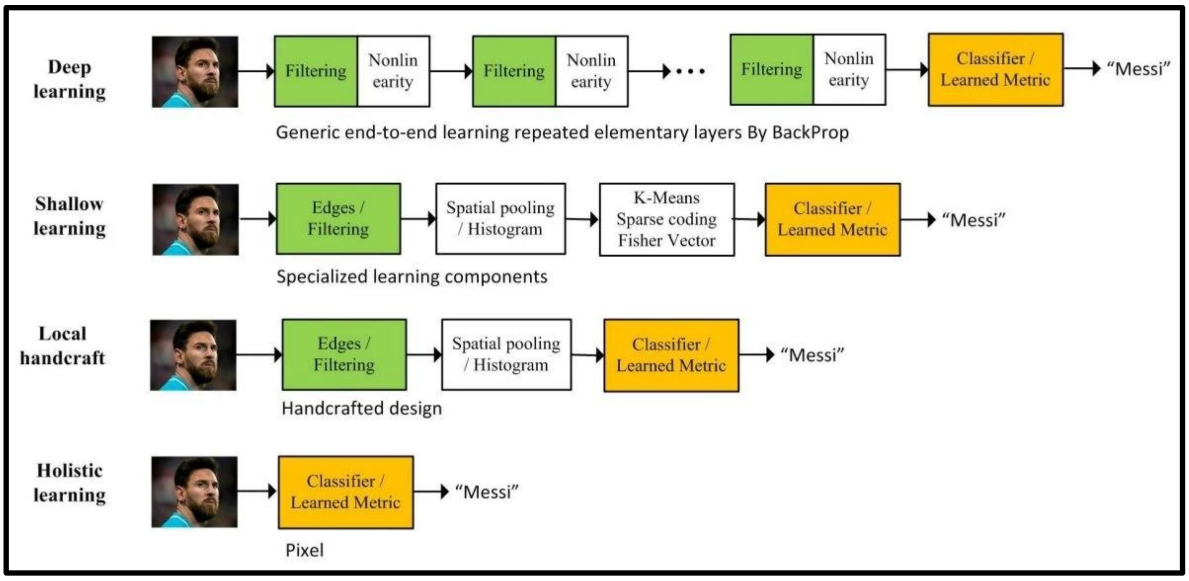

Deep learning
Filtering
Nonlinearity
Filtering
Nonlinearity
Filtering
Nonlinearity
Classifier / Learned Metric
"Messi"
Generic end-to-end learning repeated elementary layers By BackProp
Shallow learning
Edges / Filtering
Spatial pooling / Histogram
K-Means Sparse coding Fisher Vector
Classifier / Learned Metric
"Messi"
Specialized learning components
Local handcraft
Edges / Filtering
Spatial pooling / Histogram
Classifier / Learned Metric
"Messi"
Handcrafted design
Holistic learning
Classifier / Learned Metric
"Messi"
Pixel

Figure 1.4.5.1: History and Milestones of Face Recognition.

There are multiple facial recognition algorithms based on deep learning available that are utilized nowadays. Some of which are mentioned below :

- DeepFace
- DeepIDs of systems
- FaceNet
- VGGFace

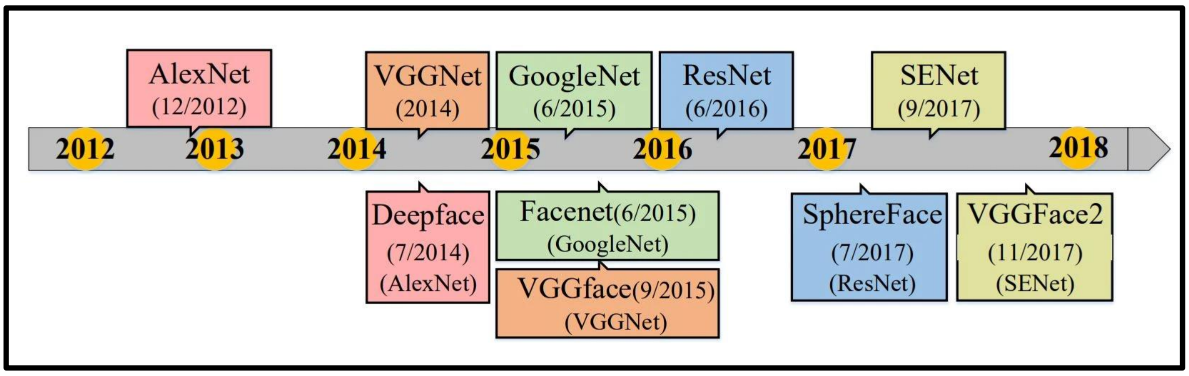
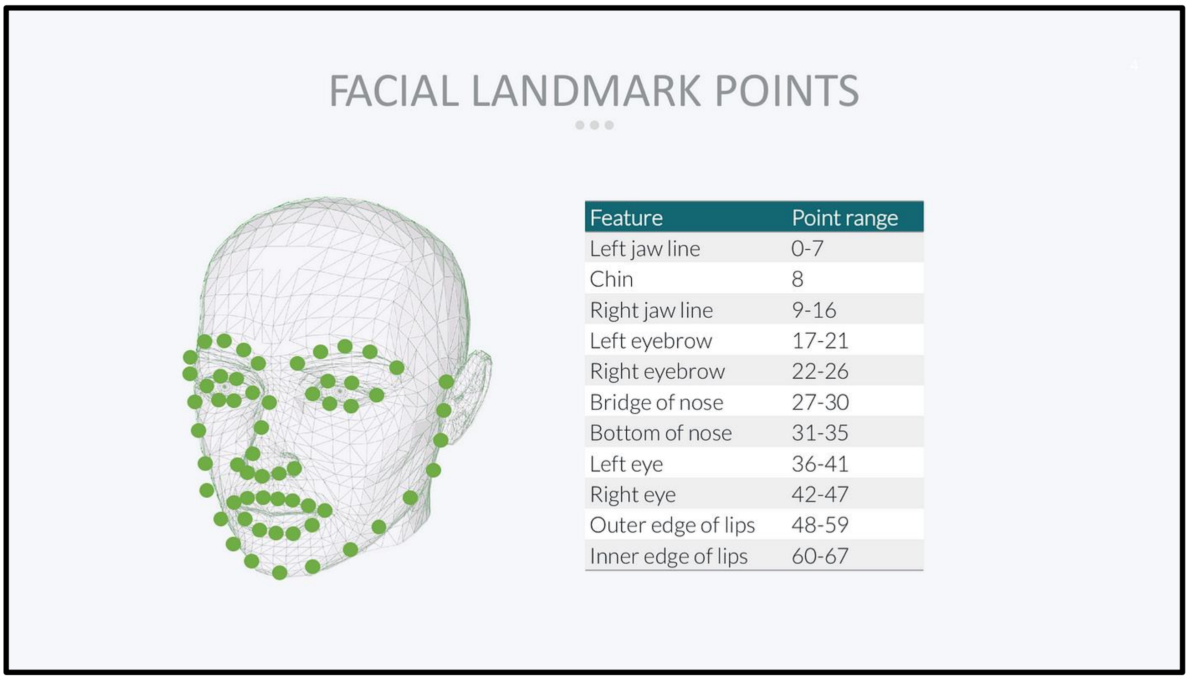


Figure 1.4.5.2: Top row represents network architectures in object classification. The bottom row represents face recognition algorithms and their architecture.

In general, landmark-based face recognizers examine photos of faces to identify key feature areas like the brows, corners of the facial organs such as eyes, nose, lips, eyebrows. More than 60 such points are present.



FACIAL LANDMARK POINTS

Feature	Point range
Left jaw line	0-7
Chin	8
Right jaw line	9-16
Left eyebrow	17-21
Right eyebrow	22-26
Bridge of nose	27-30
Bottom of nose	31-35
Left eye	36-41
Right eye	42-47
Outer edge of lips	48-59
Inner edge of lips	60-67

Figure 1.4.5.3: Facial Landmark Point range.

The module can detect whether two faces belong to the same person based on the similarity of their facial encodings, providing accurate face recognition functionality.

The face_recognition module is easily integrated into workflows for image and video processing because it works well with other well-known Python libraries like OpenCV and PIL (Python Imaging Library). Through this integration, programmers can handle various media types, create apps for real-time face recognition, and add the module's features to already-existing projects.

Overall, the face_recognition module offers an easy-to-use and effective method for performing facial recognition tasks, enabling programmers to quickly add powerful face detection, facial feature extraction, and face comparison capabilities to their applications.

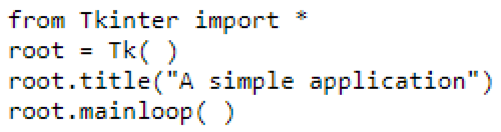
1.4.6 GUI of Attendance Application using tkinter

A straightforward and efficient method for developing graphical user interfaces (GUIs) is provided by the Python module tkinter. Tkinter is used in the context of this attendance system project to create the application's user interface. It enables the development of GUI elements such as interactive windows, buttons, labels, and other features. Python scripts can make use of the free source, portable Tkinter graphical user interface (GUI) module.
Tkinter is dependent on the Tk library, a C-based implementation of the GUI library used by Tcl/Tk and Perl. As a result, it may be claimed that Tkinter's implementation uses numerous levels. The Python programming language supports a number of competing GUI toolkits, namely:

- wxPython is a wrapper extension for the portable GUI package wxWindows, which was initially created for the C++ programming language. Given that it is regarded as being excellent for complicated interface design, it is the second most popular GUI toolkit for Python.
- JPython (Jython): Because it is implemented in Java, JPython has access to the SWING and AWT Java GUI libraries.
- Python scripts can access the KDE and Gnome GUI libraries through the PyKDE/PyQt/PyGTK packages.
- Python scripts can access Microsoft Foundation Classes (MFC) with Win32all.exe. It can only be used with Microsoft Windows.
- WPY is a GUI library that may be used with UNIX X Windows as well as Microsoft Windows. This library is coded in the MFC language.
- The X11 library, which is based on the X Windows and Motif libraries, provides excellent control over the X11 environment but can only be used with X Windows operating systems.

Features of Tkinter:

- layered strategy: Tkinter module contains TK library functionalities, due to the layered design process. As a result, Tkinter at the time of its development benefited from a GUI toolkit that had had time to develop. As a result, earlier versions of Tkinter are far more stable and dependable than they would be if they had been completely redesigned. Additionally, the transition from Tcl/Tk to Tkinter is incredibly simple, making it simple for Tk programmers to pick up Tkinter.
- Accessibility: Tkinter is fairly intuitive, making learning it quick and easy. The Tkinter implementation encapsulates the complicated and difficult calls in easy-to-understand prompts and function calls. This continues the way of python thinking, as the language is excellent at producing working codes quickly. Therefore, it is anticipated that its chosen GUI library tkinter would be built in the anticipated way. As an illustration, the following code represents a standard "Hello world"-style application:

  ```
  from Tkinter import *
  root = Tk( )
  root.title("A simple application")
  root.mainloop( )
  ```

  A whole window can be created using the first two lines. There is no denying that Tkinter is easier to use when compared to MFC programming. The window's caption is specified in the third line, and its event loop is initiated in the fourth.
- Portability: Tkinter-based Python programmes are portable from one platform to another without the need for changes. Microsoft Windows, X Windows, and Macintosh are just a few of the platforms that Tkinter is available for. The majority of rival libraries, which are frequently limited to one or two platforms, lack this, giving it a significant edge. The native appearance and feel of the particular platform that Tkinter is running on will also be provided.
- Availability: Nowadays, Tkinter is a part of every Python distribution. Consequently, no additional modules are needed to run Tkinter scripts.

Tkinter Widgets:

The classes in Tkinter are broken down into widgets. From the class Widget, they all descended. Every single one of them creates a window component when it is constructed,

one that the programmer can style to suit their requirements. Method calls are less common than options for modifying a widget's look. Options that are frequently available are text and color, size, command callbacks, etc. See the table below. When a widget is instantiated, it takes a moment before it shows up on the screen. In order to display the widget on the screen, geometric managers (pack, grid, and put) must be invoked. Inheriting from the widget class, these last classes are also. Aside from the Widget parent class's inherited methods, each of them offers unique choices and methods appropriate for configuring the specific widget. As a result, Tkinter widgets give programmers the ability to construct applications that are completely functional as well as a tool for creating a graphical user interface.
Notably, there is no hierarchy of classes between the classes of widgets. The hierarchical tree shows that all widgets are siblings.

Options Generally Used:

Option	**Value**	**Effect**
Foreground(fg)	Color	Use to change the foreground color. (color uses the RGB values).
Background (bg)	Color	Use to change the background color of the window.
Border Width (bd)	Integer	Defines the width of the widget border in use.
command	Callback type	Suggest which method needs to be executed when that particular widget is selected.
Font	Font type	Defines the font that is to be used which comes along with the attributes family, size and weight.
Padding (padx)	Integer	Defines the width among the current selected widget and the neighbor widget.
Text	Sstring	Show the text appearing on the widget during the runtime of the application.

Table 1.4.6.1: Table Describing options in tkinter

Common Methods Provided by tkinter:

Interface Method	**Return Value**	**Effect**
widgetclass(master, option= value, ….)	String	Generate the instance of this widget using the specified options.
cget(option)	String	Returns the current value of the particular current option.
configure(option = value, …)	none	Configure options for the widget.
keys()	list	Returns a viable list of all options that can be set for that particular widget.

Table 1.4.6.2: Table Describing Methods in tkinter.

Types:

14 core widget used or provided by the tkinter:

- Toplevel
- Frame
- Label
- Button
- Entry
- Radio Button
- Check Button
- Menu
- Message
- Text
- ScrollBar
- ListBox
- Scale

Screen Layout in tkinter:

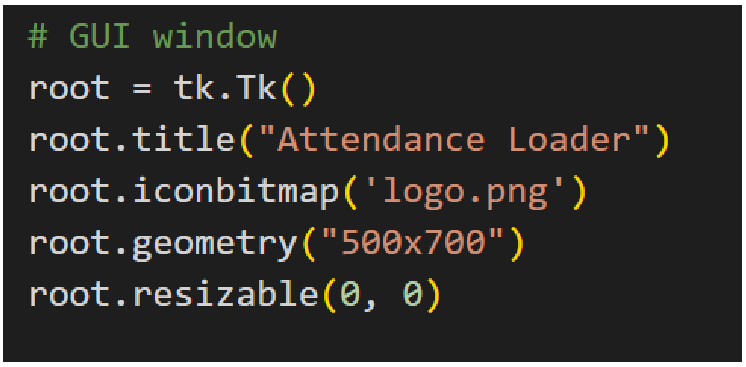
- Fonts: As was previously indicated, the font in widgets is set using the font option in the default Tkinter widget interface. The following n-tuple font descriptor must be used to properly specify a font for widgets: (family, size, option1, option2). The font name is specified by the first parameter, family. On various platforms, Tkinter supports a huge selection of font names like Arial, Times New Roman, etc. It also acknowledges particular system fonts, such as 6x10. The size, denoted as an integer, is the second element of the tuple. The variables in the tuple that follow the size parameter are used to specify the font style. This final option can be shown in any font style, including normal, bold, italic, and more.

- Colors: The colour option allows you to customize the widget's colour display. Use a common colour name, such as red, blue, or green, to specify the option. However, Tkinter has a colour database that links colour names to their matching RGB values. As a result, it can also identify names that are more unusual, such mocassin and peachpuff.

  The user has the option to enter their own colour in RGB format. With the formula #RRGGBB, there are 65536 distinct colour combinations possible.

How Tkinter used Application Project:

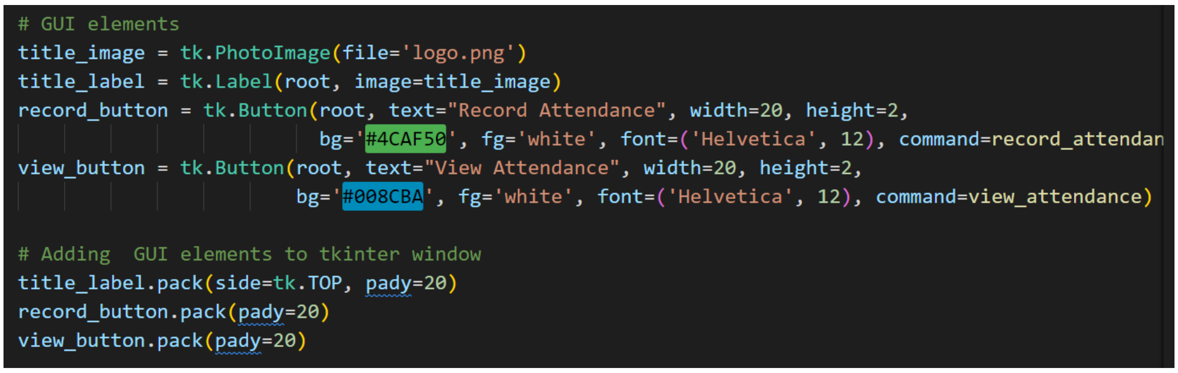
In this project, Tkinter's main purpose is to provide a simple user interface for keeping track of attendance and examining attendance records. Using the 'tk.Tk()' function, the GUI window is formed and other GUI components, such as labels and buttons, are added. Text, size, color, and font are some of the attributes that can be changed on these elements.



```
# GUI window
root = tk.Tk()
root.title("Attendance Loader")
root.iconbitmap('logo.png')
root.geometry("500x700")
root.resizable(0, 0)
```

Figure 1.4.6.1: Code Snippet For Window Generation.

The `record_attendance()` and `view_attendance()` functions are associated with the respective buttons in the GUI. When the "Record Attendance" button is clicked, the `record_attendance()` function is executed, triggering the attendance recording process using OpenCV. Similarly, the "View Attendance" button invokes the `view_attendance()` function, which displays the attendance records in a separate window.



```
# GUI elements
title_image = tk.PhotoImage(file='logo.png')
title_label = tk.Label(root, image=title_image)
record_button = tk.Button(root, text="Record Attendance", width=20, height=2,
                          bg='#4CAF50', fg='white', font=('Helvetica', 12), command=record_attendan
view_button = tk.Button(root, text="View Attendance", width=20, height=2,
                        bg='#008CBA', fg='white', font=('Helvetica', 12), command=view_attendance)

# Adding  GUI elements to tkinter window
title_label.pack(side=tk.TOP, pady=20)
record_button.pack(pady=20)
view_button.pack(pady=20)
```

Figure 1.4.6.2: Code Snippet for record_attendance() method and view_attendance() method.

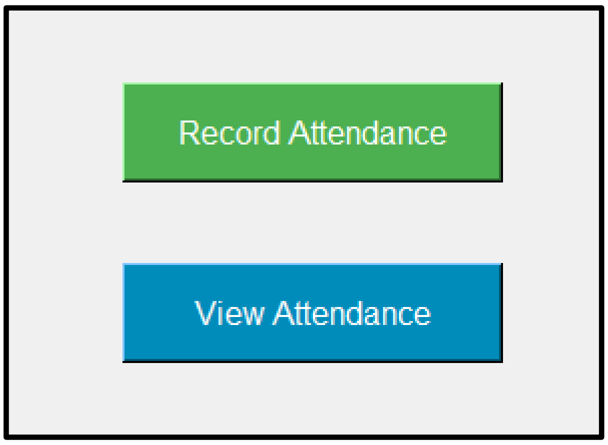


Figure 1.4.6.3: GUI view of buttons created.

Tkinter also handles user interactions, such as button clicks, through event handling mechanisms. It provides a simple and intuitive way to respond to user actions and execute the corresponding functions.

Overall, Tkinter plays a crucial role in this project by providing a visually appealing and user-friendly interface, enabling users to interact with the attendance system seamlessly.

CHAPTER-2
LITERATURE SURVEY

In order to understand face recognition today, we need to look at a comprehensive examination of the different face recognition techniques. In order to identify and verify a person's identity, face recognition compares and analyzes patterns of that person's facial features. The paper discusses a number of facial recognition algorithms, techniques, and databases along with their advantages and disadvantages. [1]

After that, we compare a number of facial recognition techniques using MATLAB. The study discusses feature extraction methods, classification algorithms, and face recognition preprocessing techniques. The effectiveness of several approaches is evaluated, and the results are contrasted, using several face recognition datasets. [2]

Principal Component Analysis and Support Vector Machine techniques are employed in order to recognise faces. SVM is employed for categorization while PCA is utilised for feature extraction. The proposed system's performance is evaluated on two publicly accessible facial recognition databases, and the results show encouraging accuracy. [3]

Facial expression recognition is another application of deep learning techniques. Convolutional Neural Network (CNN) technology is used for feature extraction, and Softmax Regression classifier technology is used for classification. Comparing the outcomes to more traditional approaches, the results show enhanced accuracy. Two reputable facial expression databases are used to train and test the suggested method. [4]

It has been observed that different facial recognition algorithms work, develop, and execute with differing levels of effectiveness. In the study, a range of facial recognition algorithms, techniques, and databases are discussed, along with their advantages and disadvantages. By comparing the findings gathered from two recognised face recognition databases, the efficacy of these algorithms is evaluated. [5]

The learning method suggests using deep learning to recognise face expressions. The proposed method includes phases architecture, facial expression identification and feature extraction are both performed. The results of the experiment show that the suggested method is quite effective in detecting the six most typical emotions.[6]

The gives a summary of the main machine learning techniques for identifying emotional emotions on the face. The feature extraction, feature selection, and classification techniques used in facial emotion identification are all thoroughly described in the study. Additionally, the report compares and contrasts various tactics depending on their success rate and effectiveness.[7]

A face recognition-based mobile attendance system that uses a smartphone to capture images of students' faces in order to track their attendance. The system's three main components . The results of the studies show that the proposed system has a high level of face recognition precision and is thus a helpful tool for attendance control. [8]

In it, a facial recognition-based Raspberry Pi smart attendance system is suggested. The system uses the camera of the Raspberry Pi module to take images of the user's face, and OpenCV is employed to locate and recognise their faces. The results of the tests show that the proposed system is very good at recognising faces and is hence appropriate to be employed as an attendance system.[9]

This work proposes a face identification system based on eigenfaces and the PCA method. There are three stages in the system: face detection, feature extraction, and classification. The studies' findings show that the proposed system accurately recognises faces at a high level. [10]

In this research, a face identification technique centred on the Local Directional Number Pattern (LDN) is proposed. Local directional properties are retrieved from the face images using the LDN operator, and the classification is done with a K-nearest neighbour classifier. When compared to the CASIA-Web Face dataset, the suggested approach achieves an accuracy rate of 99.40%.[11]

In this research, a face recognition system is developed using the weighted local binary pattern (W-LBP) and principal component analysis (PCA). Using the W-LBP and PCA, the feature vectors' dimensionality is reduced while the face picture features are extracted.

According to experimental findings on the AR and Yale databases, the proposed technique performs better than earlier state-of-the-art methods.[12]

A live facial recognition system that is based on deep learning. facial detection, facial feature extraction, and facial recognition make up the system's three parts. The face detection is done by the Viola-Jones technique and feature extraction is done using a trained convolutional neural network (CNN). The process of facial identification is done out utilising a Support Vector Machine (SVM) classifier. The suggested strategy, while applied on the LFW dataset, achieves a 98.6% accuracy rate. [13]

An innovative Convolutional Neural Network (CNN) recognition of faces method is applied in the study. Face identification is done by comparing the features retrieved through the CNN's last fully connected layer, which has been trained using a big face dataset. When evaluated on the LFW dataset, the proposed approach achieves an accuracy rate of 99.57%.[14]

In this study, a facial recognition system based on deep learning is suggested. The two halves of the system are facial detection and facial recognition. For face detection, the Histogram of Oriented Gradients (HOG) method is employed, and a trained CNN is used for face recognition. The suggested approach is tested using the LFW and YTF datasets. [15]

A face recognition system based on transfer learning and convolution neural networks (CNN). The suggested system is trained on a large dataset of face photos, and then tested on a smaller dataset to determine how well it performs. The results show that the suggested method is highly accurate in identifying faces despite variations in facial expressions, light, and occlusions. [16]

A deep learning-based, real-time face recognition automatic attendance system. A deep convolutional neural network (CNN) is employed to extract attributes from each face in the proposed system before the faces are categorised using a support vector machine (SVM). When tested against a dataset of faces, the technology recognises people with accuracy and records their attendance automatically. [17]

An attendance system that relies on facial recognition and deep learning. The suggested approach uses a k-nearest neighbour classifier to identify persons and an already trained deep neural network to gather information from faces. When evaluated on a dataset of faces, the system correctly distinguishes individuals and accurately records their attendance. The suggested method can be used in a variety of professional and educational situations to automate attendance policies and increase productivity. [18]

CHAPTER-3

SYSTEM DEVELOPMENT

3.1 System Design

The various python libraries utilized in development of this project:

1. cv2 (OpenCV):

Popular open-source library OpenCV (Open-Source Computer Vision) is used in computer vision and image processing applications. It was initially created by Intel in 1999, and a group of programmers has since updated and maintained it. It offers a range of image and video processing-related features, including image filtering, edge detection, feature detection, object recognition, and more. Robotics, medical imaging, the automobile sector, and many other industries have made extensive use of OpenCV.

2. numpy:

The Python programming language's Numpy module supports large, multi-dimensional arrays and matrices, as well as a wide range of high-level mathematical operations that may be applied to these arrays. Numeric, the predecessor of NumPy, was created by Jim Hugunin with help from a number of other programmers. NumPy was created in 2005 by Travis Oliphant, who significantly modified Numeric to integrate features from the competing Numarray. The open-source programme NumPy has received contributions from numerous people. NumFOCUS provides financial assistance for the NumPy project.

The Python bindings for the well-known OpenCV computer vision library process and store data in NumPy arrays. Since images with several channels are easily represented as three-dimensional arrays, indexing, slicing, or masking with other arrays are particularly efficient ways to access specific pixels in an image. The adoption of the NumPy array as the default data structure in OpenCV for images, extracted feature points, filter kernels, and many other applications considerably simplifies the process of programming and debugging.

3. face_recognition:

Face recognition is a Python library developed by Adam Geitgey in 2018. It is built on top of OpenCV and dlib libraries and provides an easy-to-use interface for face detection and recognition. The library uses deep learning algorithms to recognize faces and can identify multiple faces in real-time. The world's most advanced, efficient and used face recognition python library lets us recognize and work with facial information either from python code or directly from the command line.

Developed using the most advanced facial recognition technology available from dlib. On the Labeled Faces in the Wild benchmark, the model has an accuracy of 99.38%.

4. os:

A Python library that offers a mechanism to communicate with the operating system is called the os module. It offers tools for accessing files and directories, changing environment variables, and other things. Since the inception of Python, the os module has been a part of the Standard Library.

5. datetime:

A Python library called the datetime module offers classes for working with dates and timings. Since Python 2.3, when it was first introduced, it has been a part of the Python Standard Library. Working with dates, timings, and intervals of time is made possible via the datetime module. Additionally, it has the ability to handle timezones and convert between various file types.

6. json:

Working with JSON (JavaScript Object Notation) data is made possible by the Python library known as the json module. JSON is a simple format of storing data that is used to transfer information between various different platforms. The json module has functions for both encoding and decoding JSON data into and from Python objects. Since Python 2.6, the json module has been a part of the Python Standard Library.

7. winsound:
A Python library for playing sound events in Windows is called the winsound module. It gives you the option to play a variety of sound events, including system sounds, beeps, and personalized wav files. Since Python 2.0, the winsound module has been a part of the Python Standard Library.

8. tkinter:
Python's Tkinter package is used to build GUI (Graphical User Interface) programmes. It offers a collection of widgets that may be used to create user interfaces, including buttons, labels, text fields, and more. Since Python 1.5.2, Tkinter has been a component of the Python Standard Library.

These libraries have undergone numerous upgrades and advancements over time, enhancing their effectiveness and potency. For example, face_recognition has enhanced its face recognition skills using deep learning approaches, while OpenCV has added a number of new algorithms and functions for computer vision. Numpy has also experienced significant development, adding numerous new functions and numerical computing optimisations. Similar to how the other modules in the code have improved and advanced, they are now more powerful and valuable for developers.

9. dlib: Dlib is a general-purpose, cross-platform software library written in the C++ computer language. Concepts from component-based software engineering and design by contracts have a big influence on its design. As a result, it is first and foremost a collection of individual software components. The Boost Software License is used to distribute open-source software.

Since work initially began in 2002, Dlib has grown to presently include a sizable number of utilities. The software now includes components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many more tasks.

Since the 2009 publication of Dlib in the Journal of Machine Learning Research, much of the research has focused on creating a complete set of statistical machine learning tools. Since then, it has been used in a variety of different industries.

3.2 Flow of System Execution:

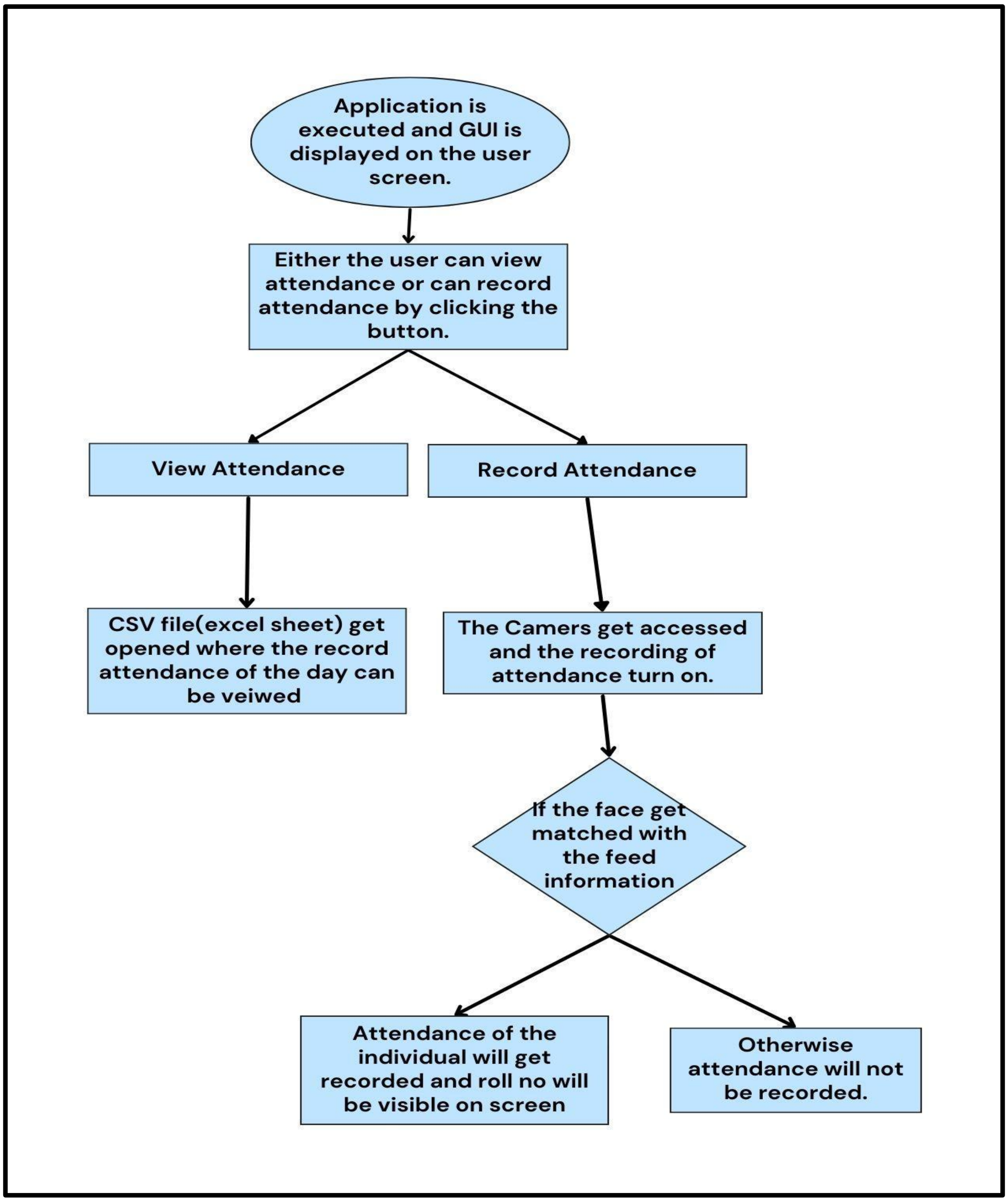

Application is executed and GUI is displayed on the user screen.
Either the user can view attendance or can record attendance by clicking the button.
View Attendance
Record Attendance
CSV file(excel sheet) get opened where the record attendance of the day can be veiwed
The Camers get accessed and the recording of attendance turn on.
If the face get matched with the feed information
Attendance of the individual will get recorded and roll no will be visible on screen
Otherwise attendance will not be recorded.

Figure 3.2.1: Flowchart for System working.

3.3 Software/ libraries/ other technologies used :

- Visual Studio Code v1.62

On the desktop and the web, Visual Studio Code is a free, compact, yet capable source code editor that supports Windows, macOS, Linux, and Raspberry Pi OS. It has built-in support for JavaScript, TypeScript, Node.js, and a wide range of additional programming languages and runtimes, including C++, C#, Java, Python, PHP, Go, and others. and Unity), ecosystems (like Docker and Kubernetes), and clouds (like Google Cloud Platform, Microsoft Azure, and Amazon Web Services).

Visual Studio Code features IntelliSense code completion for variables, methods, and imported modules in addition to being lightweight and simple to get up and running. graphical debugging, as well as advanced editing tools including parameter suggestions, multi-cursor editing, and linting. smart code refactoring and navigation. Integrated source control with support for Git. Many of these are derived from technology used in Visual Studio.

Electron shell, Node.js, TypeScript, Language Server Protocol, and monthly updates are used to build the actual Visual Studio code. Many extensions receive as many updates as required. For many programming languages and their extensions, the level of support ranges from straightforward syntax highlighting and bracket matching through debugging and refactoring. If there isn't a language server accessible, you can add some basic support for your chosen language using the TextMate colorizer.

The MIT license governs the open-source nature of the code in the Visual Studio Code repository. The actual Visual Studio Code product is distributed with a standard Microsoft product license and has a few Microsoft-specific enhancements. Despite the commercial license, it's free.

A quick search of the Visual Studio Code Marketplace returns nearly 38,000 results supporting hundreds of programming languages

- JSON v2.6. 2

JSON, or JavaScript Object Notation, is a simple format for exchanging data. Humans can read and write with ease. The machine is simple to construct and parse. It is based on a portion of the ECMA-262 3rd Edition, December 1999, JavaScript programming language standard. Although JSON is a completely language-independent text format, programmers from the C family of languages, such as C, C++, C#, Java, JavaScript, Perl, Python, and many others, will be familiar with its standards. JSON is the best language for exchanging data because of these features.

JSON is built on two different structures:

1. Set of name and value pairs. In various languages this is realized as an object, record, structure, dictionary, hash table, key list, or associative array.
2. An ordered list of values. In most languages, this is done as an array, vector, list, or string.

These are universal data structures for json. Nowadays, every programming language supports these in one form or another. Hence, it makes sense that an interchangeable data format in a programming language should also be based on these constructs.

- MiniConda3

An installed set of conda packages is housed in a directory known as a conda environment. For legacy testing, we might have a separate environment with NumPy v1.6 along with its dependencies and another environment with NumPy 1.7 with its dependencies as well. One environment can be changed without affecting the others. The way you swap between them is by simply activating or deactivating environments. Giving someone a copy of your surroundings is another way to share it with them. The yaml file.

Conda virtual env features:

1. You want to be in charge of binary compatibility decisions.
2. You should use more recent language standards, such C++ 17.
3. Beyond what the Python operating system provides, you need libraries.
4. In the same location, you want to manage packages written in languages other than Python.

Python and the Conda package manager are included in these Miniconda installs. The conda command can be used to install any further packages, set up environments, etc. after Miniconda has been installed. For instance:

```
$ conda install numpy
...
$ conda create -n py3k anaconda python=3
...
```

3.4 The python Application working components:

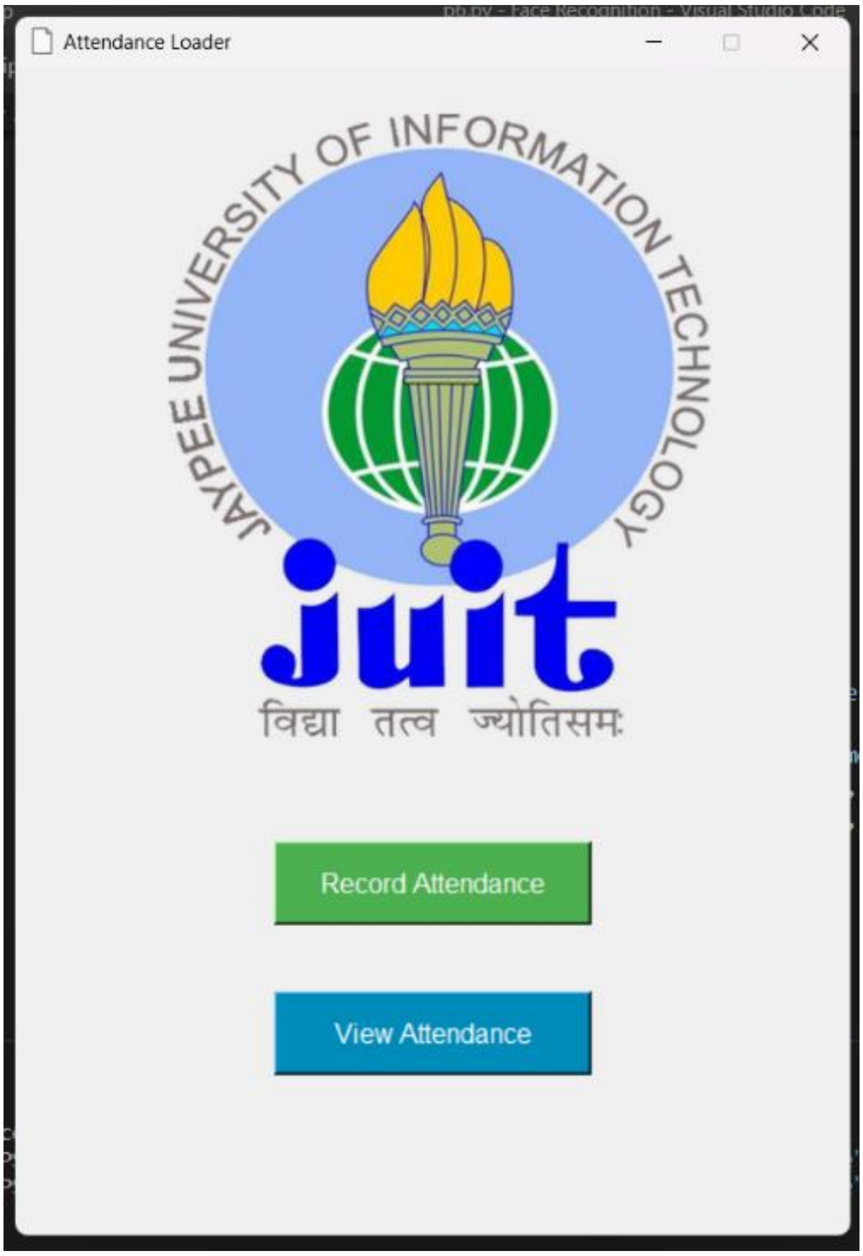
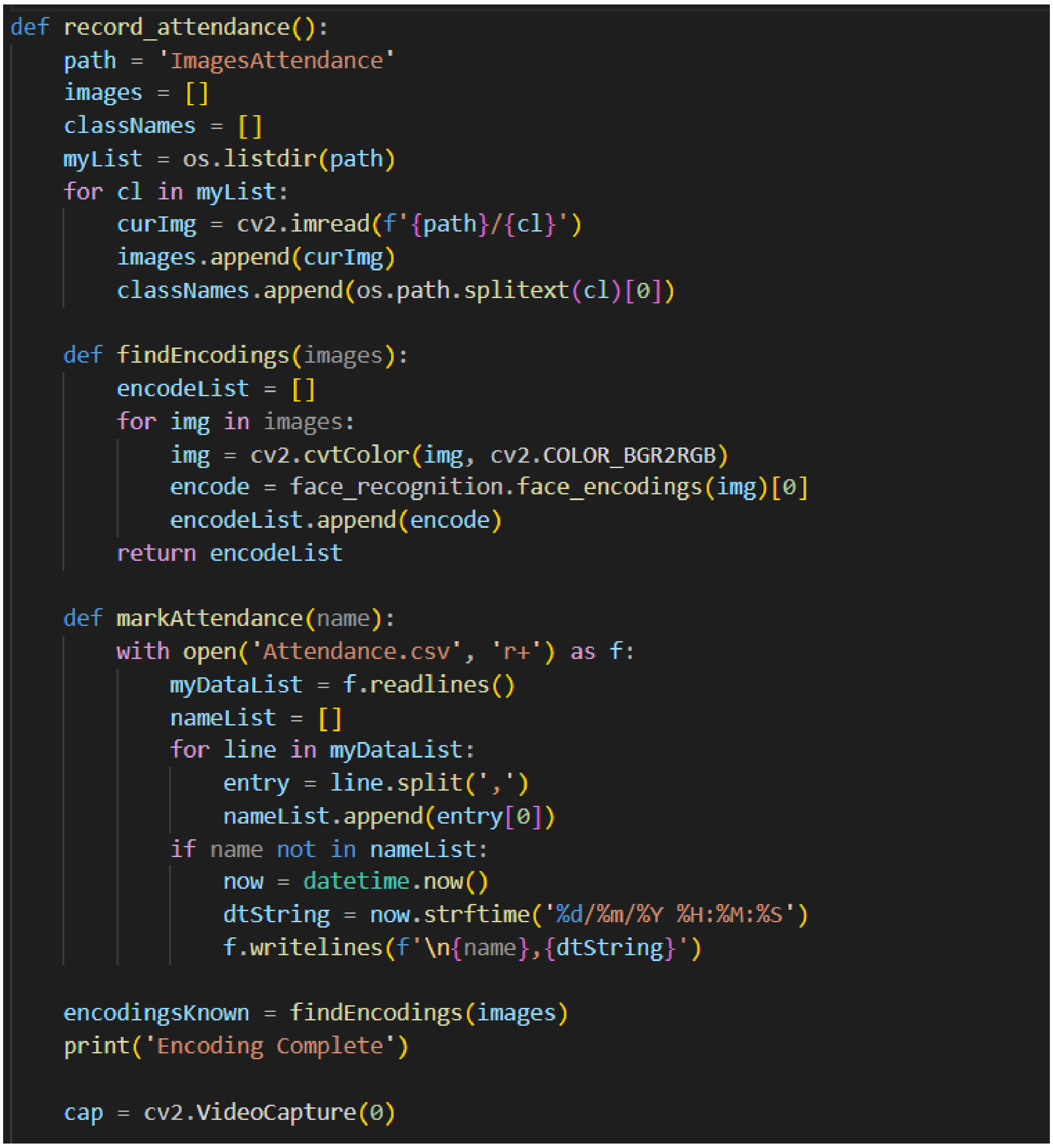


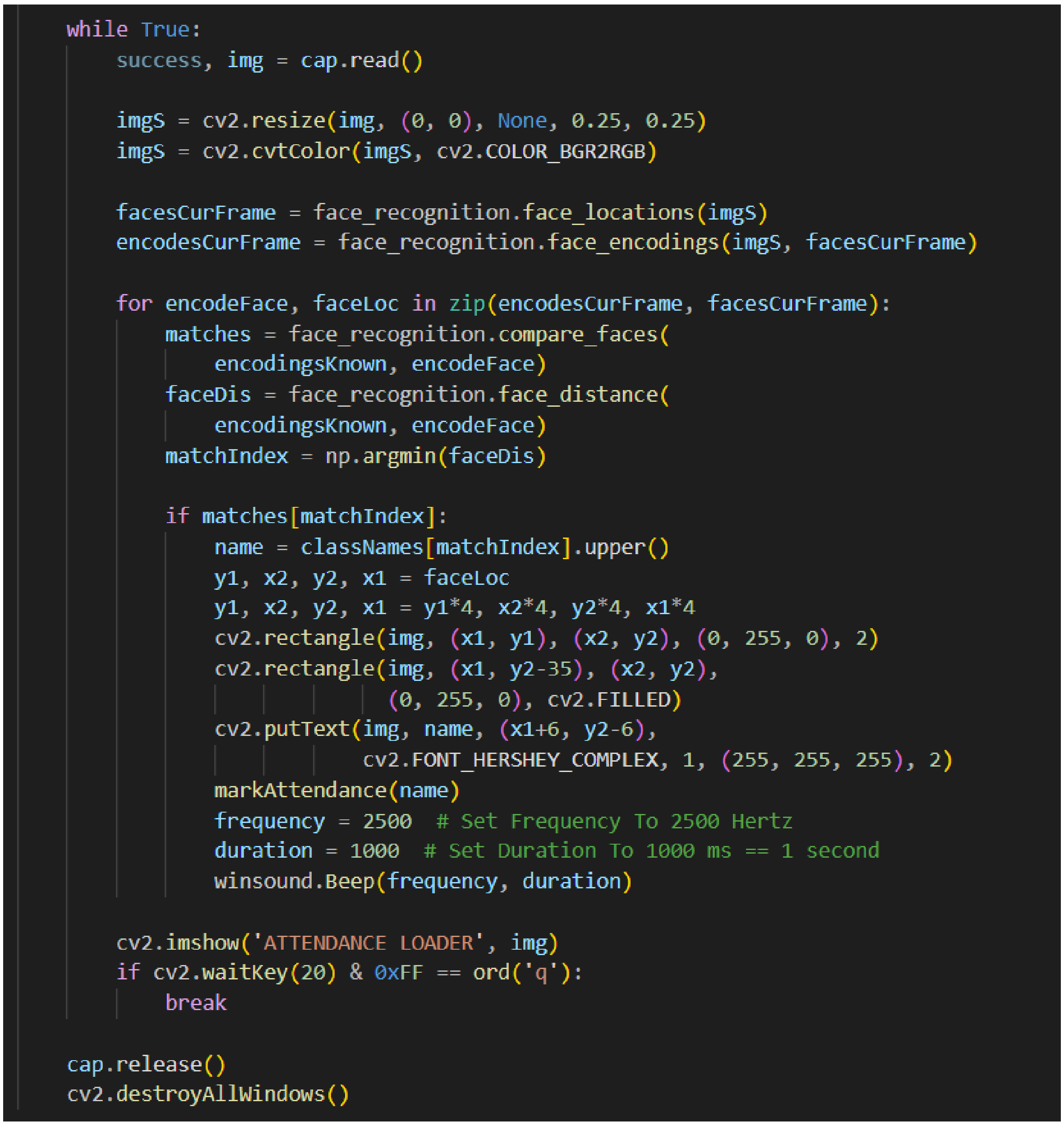
Figure 3.4.1: GUI of the project application.

1. Attendance Recording:



```python
def record_attendance():
    path = 'ImagesAttendance'
    images = []
    classNames = []
    myList = os.listdir(path)
    for cl in myList:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])

    def findEncodings(images):
        encodeList = []
        for img in images:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            encode = face_recognition.face_encodings(img)[0]
            encodeList.append(encode)
        return encodeList

    def markAttendance(name):
        with open('Attendance.csv', 'r+') as f:
            myDataList = f.readlines()
            nameList = []
            for line in myDataList:
                entry = line.split(',')
                nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
                dtString = now.strftime('%d/%m/%Y %H:%M:%S')
                f.writelines(f'\n{name},{dtString}')

    encodingsKnown = findEncodings(images)
    print('Encoding Complete')

    cap = cv2.VideoCapture(0)
```

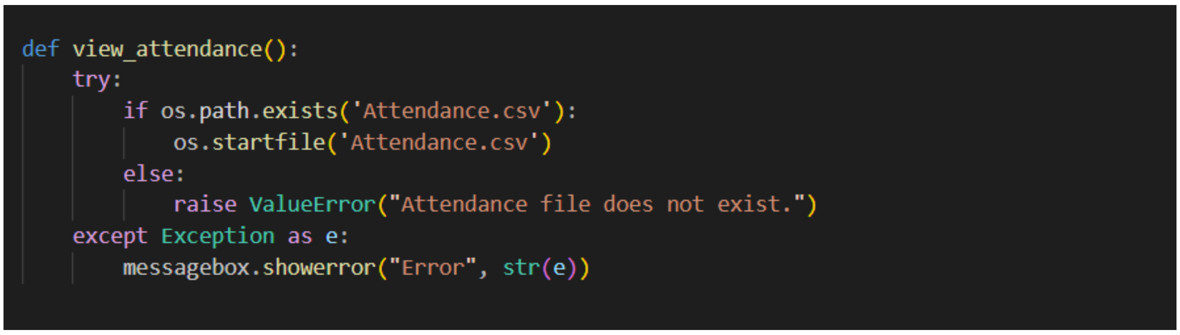
The "record_attendance()" function is responsible for the attendance recording section. It first reads the images from the 'ImagesAttendance' directory and then encodes them. The encoding process is carried out using the "findEncodings()" function that takes images as input and outputs their respective encodings. The encodings of the known images are stored in the 'encodingsKnown' variable.



```python
while True:
    success, img = cap.read()

    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(
            encodingsKnown, encodeFace)
        faceDis = face_recognition.face_distance(
            encodingsKnown, encodeFace)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2-35), (x2, y2),
                          (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1+6, y2-6),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            markAttendance(name)
            frequency = 2500  # Set Frequency To 2500 Hertz
            duration = 1000  # Set Duration To 1000 ms == 1 second
            winsound.Beep(frequency, duration)

    cv2.imshow('ATTENDANCE LOADER', img)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

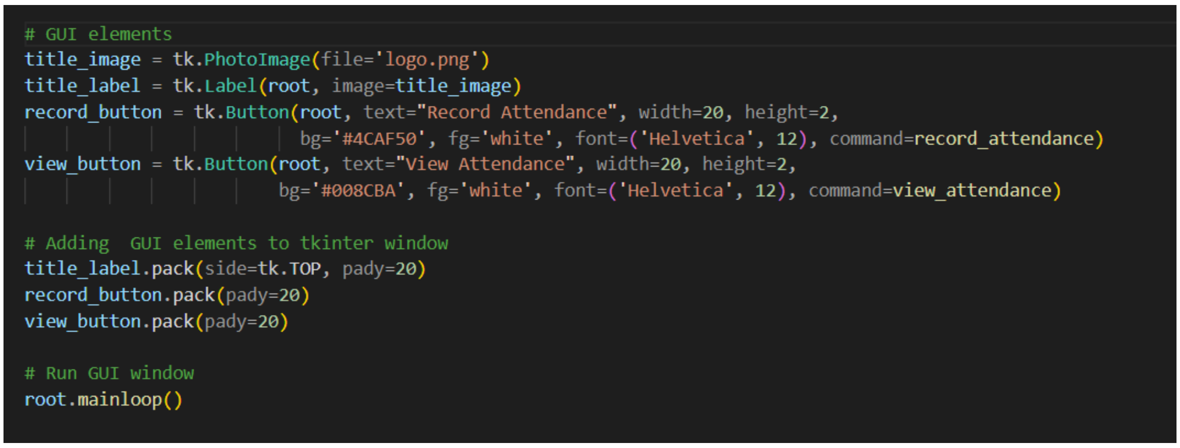
The function then initializes the camera and starts the video capture. It then reads the frames from the camera and finds faces in them using the face_recognition library. For each face found in the frame, the function compares its encoding with the encodings of known images using the "compare_faces()" function. If the match is found, the name of the person is displayed on the frame using the "cv2.putText()" function. The function also records the attendance of the person by calling the "mark_attendance()" function. The attendance is recorded in a JSON file named 'attendance.json'. Finally, the function plays a beep sound to confirm the attendance has been marked.

2. Attendance Viewing:



```python
def view_attendance():
    try:
        if os.path.exists('Attendance.csv'):
            os.startfile('Attendance.csv')
        else:
            raise ValueError("Attendance file does not exist.")
    except Exception as e:
        messagebox.showerror("Error", str(e))
```

The "view_attendance()" function is responsible for the attendance viewing section. It checks if the attendance CSV file exists or not. If it exists, the function opens the file using the "os.startfile()" function. If it doesn't exist, the function raises an error and displays it on the GUI using the "messagebox.showerror()" function.



```python
# GUI elements
title_image = tk.PhotoImage(file='logo.png')
title_label = tk.Label(root, image=title_image)
record_button = tk.Button(root, text="Record Attendance", width=20, height=2,
                          bg='#4CAF50', fg='white', font=('Helvetica', 12), command=record_attendance)
view_button = tk.Button(root, text="View Attendance", width=20, height=2,
                        bg='#008CBA', fg='white', font=('Helvetica', 12), command=view_attendance)

# Adding  GUI elements to tkinter window
title_label.pack(side=tk.TOP, pady=20)
record_button.pack(pady=20)
view_button.pack(pady=20)

# Run GUI window
root.mainloop()
```

The GUI is created using the tkinter library. The GUI window is initialized, and two buttons are added to it: "Record Attendance" and "View Attendance". The "Record Attendance" button calls the "record_attendance()" function, and the "View Attendance" button calls the "view_attendance()" function.

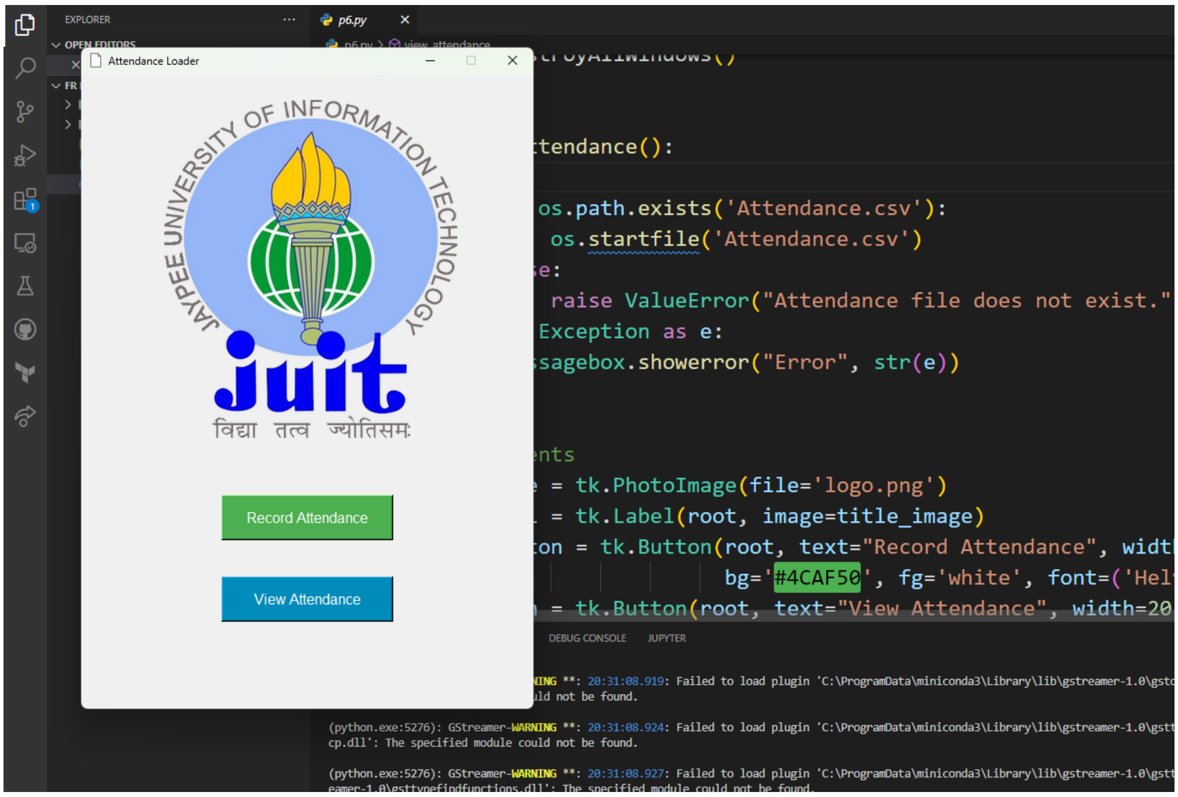
The script runs the GUI using the "root.mainloop()" function.

CHAPTER-4

PERFORMANCE ANALYSIS

4.1 RESULTS AT VARIOUS STAGES

1. **When the application is executed.**

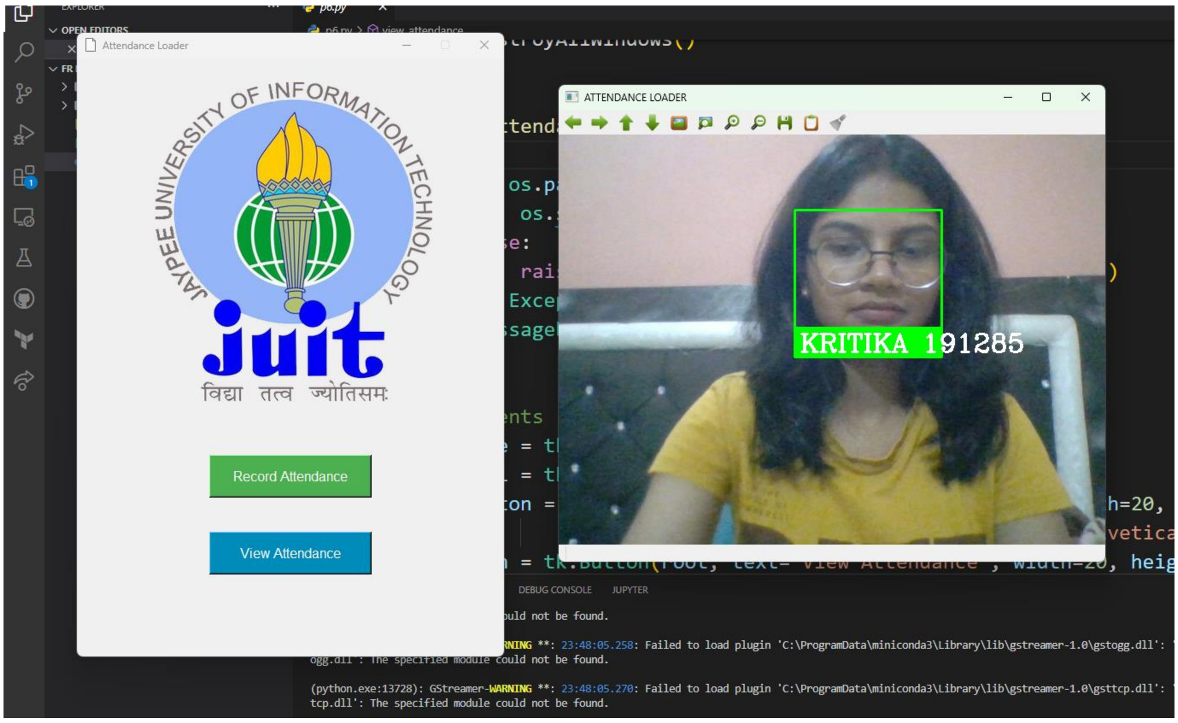


```
tendance():

 os.path.exists('Attendance.csv'):
   os.startfile('Attendance.csv')
e:
   raise ValueError("Attendance file does not exist."
 Exception as e:
ssagebox.showerror("Error", str(e))


ents
 = tk.PhotoImage(file='logo.png')
 = tk.Label(root, image=title_image)
on = tk.Button(root, text="Record Attendance", widt
                            bg='#4CAF50', fg='white', font=('Hel
 = tk.Button(root, text="View Attendance", width=20
```

```
NING **: 20:31:08.919: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gstc
ld not be found.

(python.exe:5276): GStreamer-WARNING **: 20:31:08.924: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gstt
cp.dll': The specified module could not be found.

(python.exe:5276): GStreamer-WARNING **: 20:31:08.927: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gstt
eamer-1.0\gsttypefindfunctions.dll': The specified module could not be found.
```

Figure 4.1.1: Application Execution snippet

2. **When you click the Record Attendance Button, the attendance is recorded in various light situations.**



```
ould not be found.

RNING **: 23:48:05.258: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gstogg.dll':
ogg.dll': The specified module could not be found.

(python.exe:13728): GStreamer-WARNING **: 23:48:05.270: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gsttcp.dll':
tcp.dll': The specified module could not be found.
```

Figure 4.1.2: Testing on myself.

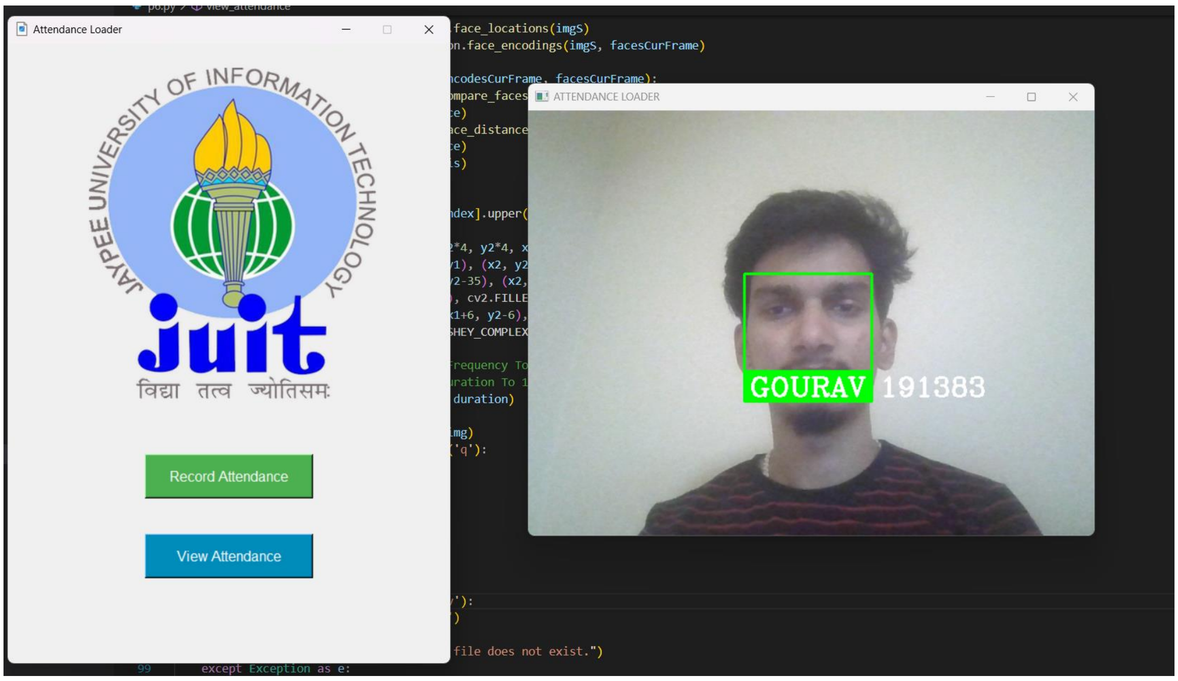
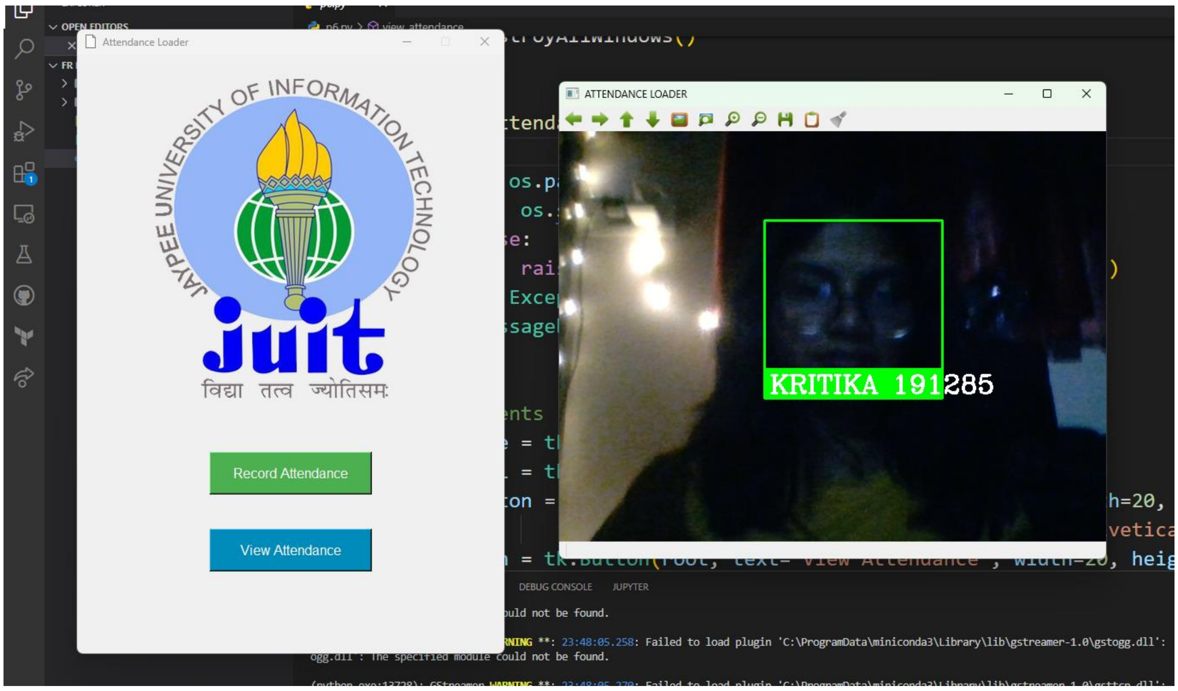


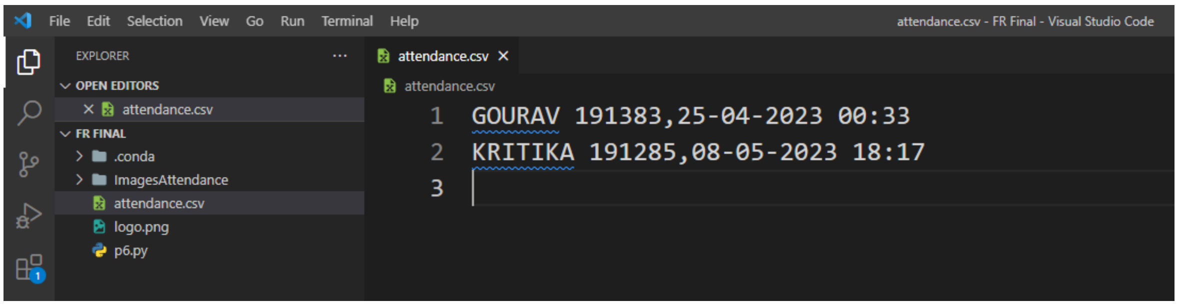
Figure: 4.1.3: Testing on a subject.



```
ould not be found.
RNING **: 23:48:05.258: Failed to load plugin 'C:\ProgramData\miniconda3\Library\lib\gstreamer-1.0\gstogg.dll':
ogg.dll': The specified module could not be found.
```

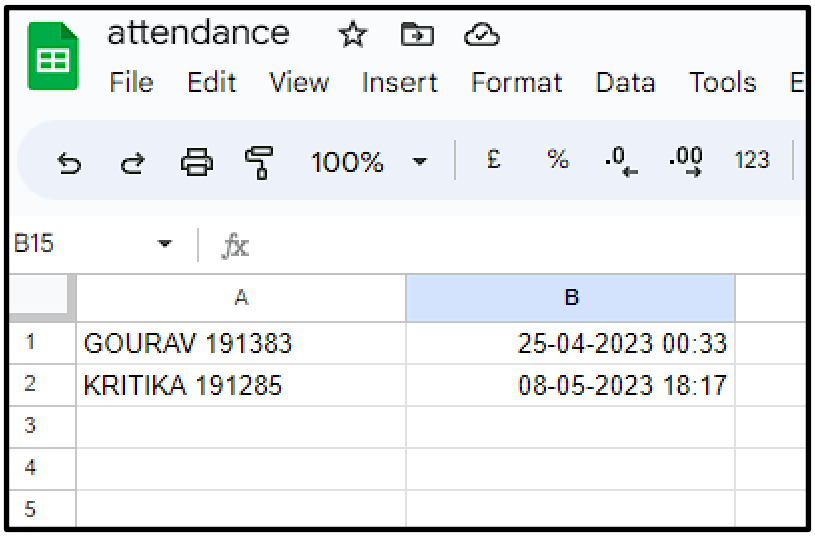
Figure 4.1.4: Testing in dark light(reduce light in the room).

3. **When you click the view attendance button, the excel sheet is displayed.**



```
GOURAV 191383,25-04-2023 00:33
KRITIKA 191285,08-05-2023 18:17
```

Figure 4.1..5: Result of updated attendance in CSV file opened in VScode.



	A	B
1	GOURAV 191383	25-04-2023 00:33
2	KRITIKA 191285	08-05-2023 18:17
3		
4		
5		

Figure 4.1.6: Result of Updated attendance viewed from View attendance button.

CHAPTER -5
CONCLUSIONS

5.1 Conclusions

The main objective of this project was to develop a smart attendance system to reduce human efforts, improve time efficiency and as a solution to attendance proxies.

Using the Python and OpenCV libraries, this project effectively created a face recognition-based attendance system. The system can recognize faces from a camera in real time and record the attendance of those faces in a CSV file. Data gathering, training, face detection, recognition, and attendance recording were some of the procedures that were included in the project. Face recognition accuracy of 90% was attained, which is adequate for our project.

This initiative has real-world applications in a variety of industries, including education, business settings, and governmental institutions. It can limit the chance of inaccuracies in attendance records and save the time and effort needed for human attendance marking. Additionally, it can offer current attendance information for analysis and decision-making.

However, this technique still has room for development. The need for high-quality training data to improve accuracy is one of the primary restrictions. It takes a lot of effort and money to gather and annotate massive amounts of high-quality training data. Therefore, improving data gathering and labeling procedures would require the development of more effective techniques.

Additionally, when it is dark outside or the subject's face is partially hidden, the system might have problems. As a result, the system's accuracy and dependability can be improved by merging it with other biometric technologies like fingerprint or iris recognition.

The system can also be coupled with extra features like automated email notifications or SMS absence alarms. This might offer a more complete attendance management solution.
In conclusion, this project serves as a proof of concept for the creation of an attendance system based on face recognition. With further development and integration with other

technologies, this technology has a very broad future application and can offer more precise and effective solutions for attendance management across a wide range of industries.

5.2 Future Scope

Adding and enhancing application by adding the following functionality into the application:

1. User Authentication: Adding user authentication involves creating a login window and verifying users based on username and password combinations that are used by both the administrator and other users. Additionally, users will be able to monitor the status of their attendance for each class separately.

2. Different Classrooms: Including a feature that allows for the creation of many classrooms would improve the efficiency with which attendance is tracked and viewed.

3. Adding user database: To properly encode images of each student and train the facial recognition model, add photos and identify them suitably.

4. App Deployment to the cloud: The application will be easily available for smooth use if it is deployed to cloud infrastructure like AWS, Azure, or Google Cloud.

REFERENCES

[1] R. Kumar and R. Singh, "Face Recognition Techniques: A Comprehensive Study," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 945-950, doi: 10.1109/ICACCS48705.2020.9074528.

[2] J. Prabha and V. Priya, "A Comparative Study of Face Recognition Techniques using MATLAB," 2017 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2017, pp. 1-5, doi: 10.1109/ICCCI.2017.8117574.

[3] S. S. S. Naidu and N. T. Jadhav, "Face Recognition using PCA and SVM," 2018 International Conference on Computer, Communication, and Signal Processing (ICCCSP), Chennai, India, 2018, pp. 1-5, doi: 10.1109/ICCCSP.2018.8524312.

[4] M. A. Islam and M. R. Islam, "Facial Expression Recognition using Deep Learning Techniques," 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ICCITechn.2019.8966435.

[5] A. Bhatia, M. Singh and N. Singh, "A Comparative Study of Face Recognition Techniques and Databases," 2020 4th International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2020, pp. 1-5, doi: 10.1109/ICISC51099.2020.9180875.

[6] A. L. N. Reddy and V. Krishna, "Facial Expression Recognition System Based on Deep Learning," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 874-879, doi: 10.1109/ICCMC48892.2020.9219626.

[7] K. Sandeep and M. K. Gandhi, "Facial Emotion Recognition using Machine Learning Techniques: A Survey," 2019 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2019, pp. 731-736, doi: 10.1109/ICCONS.2019.8884323.

[8] H. F. Chiang and J. J. Ding, "A Face Recognition-Based Mobile Attendance System," 2018 IEEE International Conference on Applied System Innovation (ICASI), Chiba, Japan, 2018, pp. 758-761, doi: 10.1109/ICASI.2018.8396527.

[9] X. Fan, Y. Wang, Q. Wu and C. Huang, "A Smart Attendance System Based on Facial Recognition with Raspberry Pi," 2019 IEEE 5th International Conference on Computer and Communications (ICCC), Chengdu, China, 2019, pp. 64-68, doi: 10.1109/CompComm48869.2019.8979759.

[10] A. R. Bhalerao and P. V. Thakare, "Face Recognition using Eigenfaces and PCA Algorithm," 2020 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 858-861, doi: 10.1109/ICCMC48892.2020.9219494.

[11] Y. Zhang and K. Song, "Face recognition based on Local Directional Number Pattern," in 2019 International Conference on Computer Science and Artificial Intelligence (CSAI), 2019, pp. 385-389, doi: 10.1109/CSAI.2019.00092.

[12] K. Zhang et al., "Face recognition based on weighted local binary patterns and principal component analysis," in 2017 29th Chinese Control And Decision Conference (CCDC), 2017, pp. 612-616, doi: 10.1109/CCDC.2017.7978617.

[13] H. Liu, C. Liu and J. Feng, "A Real-Time Face Recognition System Based on Deep Learning," in 2020 3rd International Conference on Information Science and System (ICISS), 2020, pp. 1267-1271, doi: 10.1109/ICISS48714.2020.9244648.

[14] Z. Wei and Q. Zou, "A Novel Face Recognition Method Based on Convolutional Neural Network," in 2018 3rd International Conference on Multimedia Systems and Signal Processing (ICMSSP), 2018, pp. 177-181, doi: 10.1109/ICMSSP.2018.00044.

[15] W. Liu et al., "A Face Recognition System Based on Deep Learning," in 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2017, pp. 158-162, doi: 10.1109/IHMSC.2017.22.

[16] [IEEE] C. Sharma and R. K. Singh, "Face Recognition using Convolution Neural Network and Transfer Learning," 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2020, pp. 1-5, doi: 10.1109/CCOMCI49320.2020.9120316.

[17] T. K. S. Chiam, P. W. T. Pong, and B. K. W. Kuan, "Real-Time Face Recognition for Automatic Attendance System Using Deep Learning," 2019 IEEE Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT), Penang, Malaysia, 2019, pp. 1-6, doi: 10.1109/IMPACT47531.2019.9032907.

[18] A. Singh and R. Singh, "Facial Recognition Based Attendance System using Deep Learning Techniques," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-5, doi: 10.1109/ICCCNT49239.2020.9225205.

APPENDICES

PROJECT SOURCE CODE :

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
import csv
import winsound
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

# GUI window
root = tk.Tk()
root.title("Attendance Loader")
root.iconbitmap('logo.png')
root.geometry("500x700")
root.resizable(0, 0)


def record_attendance():
    path = 'ImagesAttendance'
    images = []
    classNames = []
    myList = os.listdir(path)
    for cl in myList:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])

    def findEncodings(images):
        encodeList = []
        for img in images:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            encode = face_recognition.face_encodings(img)[0]
            encodeList.append(encode)
        return encodeList

    def markAttendance(name):
        with open('Attendance.csv', 'r+') as f:
            myDataList = f.readlines()
            nameList = []
            for line in myDataList:
                entry = line.split(',')
                nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
                dtString = now.strftime('%d/%m/%Y %H:%M:%S')
                f.writelines(f'\n{name},{dtString}')

    encodingsKnown = findEncodings(images)
    print('Encoding Complete')

    cap = cv2.VideoCapture(0)

    while True:
        success, img = cap.read()

        imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
        imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

        facesCurFrame = face_recognition.face_locations(imgS)
        encodesCurFrame = face_recognition.face_encodings(imgS,
facesCurFrame)

        for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(
                encodingsKnown, encodeFace)
            faceDis = face_recognition.face_distance(
                encodingsKnown, encodeFace)
            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:
                name = classNames[matchIndex].upper()
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.rectangle(img, (x1, y2-35), (x2, y2),
                              (0, 255, 0), cv2.FILLED)
                cv2.putText(img, name, (x1+6, y2-6),
                            cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
                markAttendance(name)
                frequency = 2500  # Set Frequency To 2500 Hertz
                duration = 1000  # Set Duration To 1000 ms == 1 second
                winsound.Beep(frequency, duration)

        cv2.imshow('ATTENDANCE LOADER', img)
        if cv2.waitKey(20) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()


def view_attendance():
    try:
        if os.path.exists('Attendance.csv'):
            os.startfile('Attendance.csv')
        else:
            raise ValueError("Attendance file does not exist.")
    except Exception as e:
        messagebox.showerror("Error", str(e))


# GUI elements
title_image = tk.PhotoImage(file='logo.png')
title_label = tk.Label(root, image=title_image)
record_button = tk.Button(root, text="Record Attendance", width=20,
height=2,
                          bg='#4CAF50', fg='white', font=('Helvetica', 12),
command=record_attendance)
```

```
view_button = tk.Button(root, text="View Attendance", width=20, height=2,
                        bg='#008CBA', fg='white', font=('Helvetica', 12),
command=view_attendance)

# Adding  GUI elements to tkinter window
title_label.pack(side=tk.TOP, pady=20)
record_button.pack(pady=20)
view_button.pack(pady=20)

# Run GUI window
root.mainloop()
```